Article

Facilitating Machine Learning Model Comparison and Explanation Through A Radial Visualisation*

Jianlong Zhou, Weidong Huang and Fang Chen

University of Technology Sydney, Sydney, Australiia; {firstName.lastName}@uts.edu.au
* Correspondence: jianlong.zhou@uts.edu.au, weidong.huang@uts.edu.au

Abstract: Building an effective Machine Learning (ML) model for a data set is a difficult task involving various steps. One of the most important steps is to compare generated substantial amounts of ML models to find the optimal one for the deployment. It is challenging to compare such models with dynamic number of features. Comparison is more than just finding differences of ML model performance, users are also interested in the relations between features and model performance such as feature importance for ML explanations. This paper proposes *RadialNet Chart*, a novel visualisation approach to compare ML models trained with a different number of features of a given data set while revealing implicit dependent relations. In RadialNet Chart, ML models and features are represented by lines and arcs respectively. These lines are generated effectively using a recursive function. The dependence of ML models with dynamic number of features is encoded into the structure of visualisation, where ML models and their dependent features are directly revealed from related line connections. ML model performance information is encoded with colour and line width in RadialNet Chart. Together with the structure of visualisation, feature importance can be directly discerned in RadialNet Chart for ML explanations. Compared with other commonly used visualisation approaches, RadialNet Chart can help to simplify the ML model comparison process with different benefits such as: more efficient to help users focus their attention to find visual elements of interest, easier to compare ML performance to find optimal ML model and discern important features visually and directly, instead of through complex algorithmic calculations for ML explanations.

Keywords: Machine learning; performance; bar chart; line chart; radar chart; RadialNet chart; visualisation

Citation: Zhou, J.; Huang, W.; Chen, F. Facilitating Machine Learning Model Comparison and Explanation Through A Radial Visualisation. *Energies* **2021**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

1. Introduction

We have witnessed a rapid boom of data in recent years from various fields such as infrastructure, transport, energy, health, education, telecommunications, and finance. Together with the dramatic advances in Machine Learning (ML), getting insights from these "Big Data" and data analytics-driven solutions are increasingly in demand for different purposes. While these "Big Data" are used by sophisticated ML algorithms to train ML models which are then evaluated by various metrics such as accuracy, the generated substantial amounts of ML models must be compared by the engineering designers and analysts to find the optimal one for the deployment. Fig. 1 shows a typical pipeline that processes data to find an optimal ML model. Taking a data set with multiple features for ML training as an example, multiple features can be grouped differently as the input for an ML algorithm to train different ML models. For example, if a data set has three features of F1, F2, and F3, these features may have seven different groups: [F1], [F2], [F3], [F1, F2], [F1, F3], [F2, F3], and [F1, F2, F3]. Each feature group can be used as the input for an ML algorithm to train an ML model, thereby obtaining seven different ML models. It is a common thread to find the best/worst model by comparing such models, however it is often challenging when having a large number of features. Furthermore, comparison is more than just finding differences of ML model performance, users are also interested in the relations between features and model performance from comparison to get explanation of models, for example, to find which features result in high performance of ML models, and those features are referred

*This paper is an extended version of our paper published in 2020 IEEE Pacific Visualization Symposium (PacificVis), Tianjin, China, 3-5 June 2020; pp.226-230.

as high important features, or vice versa. This is because the identification of the most or least important features are the key steps for feature engineering in effective and explainable machine learning.

On the other hand, it is widely recognised that visualisations amplify human's cognition during data analysis [1] and proper visualisation of ML outcomes is essential for a human analyst to be able to interpret them [2–4]. Viegas and Wattenberg [5] claimed that "data visualisation of the performance of algorithms for the purpose of identifying anomalies and generating trust is going to be the major growth area in data visualisation in the coming years". More importantly, comparison with visualisation is imperative to identify the optimal model from substantial amounts of ML models. Bar chart, radar chart, line chart as well as others [6] are commonly used visualisation methods in machine learning to compare different variables. However, comparison of ML models with a large number of features is still considered challenging with the aid of these commonly used visualisations: the items for comparison and the relationships between them can be highly complicated. While these commonly used visualisation approaches not only cause information clutters for large number of visual elements (e.g. bars, dots, lines) but also miss relation information between features and models, which are significant in ML explanations. It is also very difficult for users to differentiate differences of various model performances with these commonly used visualisation approaches. Despite the specific focus on visualising comparison in recent studies [7–10], little work has been done on the visual comparison of ML models while identifying relations between features and ML models (e.g. the most and least important features).

We explore an approach based on the structure of visualisation in addressing challenges of comparison ML models with dynamic number of features: while height information of bars and lines in commonly used visualisation approaches only encode one-dimensional information in a 2-dimensional (2D) space, it is possible to encode ML model information in other dimensions of the space. If both visual elements and structure of visualisation can be used to encode information of ML models, insights about ML models could be automatically generated, users would not have to inspect every model to find optimal one or conduct complex calculations to estimate feature importance.

In this paper, we propose *RadialNet Chart* (also referred to RadialNet in this paper), a novel visualisation approach to compare ML models with different number of features while revealing implicit dependent relations. In RadialNet, ML models and features are represented by lines and arcs respectively (an arc also represents the model based on the single feature of arc). The challenge of revealing dependence of ML models with dynamic number of features is addressed by encoding such information into the structure of visualisation, where ML models and their dependent features are directly revealed from related line connections. These lines are defined using a recursive function to generate them effectively. ML model performance information is encoded with colour and line width in RadialNet. It simplifies the comparison of different ML models based on these visual encoding. Moreover, together with the structure of visualisation, feature importance can be directly discerned in RadialNet for ML explanation. RadialNet uses a concept of feature path for ML model lines to avoid a large number of line entangles. And when visual elements for ML models are crowded, RadialNet allows to interactively change spanning space that RadialNet covers to dynamically control the visual complexities. To understand the effectiveness of RadialNet, we conducted a comparison experiment with three commonly used visualisation approaches of line chart, bar chart, and radar chart. The comparison experiment was evaluated with eleven researchers and developers experienced in machine learning related areas. The findings show that RadialNet has advantages in identifying features related to specific models as well as directly revealing importance of features (for ML explanations). Furthermore, RadialNet is more efficient to help users focus their attention to find visual elements of interest. It is more compact to show more information in a limited space compared with other visualisation types.

This paper is the extended version of the conference paper of [10]. This extended version includes a detailed literature review with more related works, and more detailed information about the methodology. Since the complexity of RadiaNet, this extended version provides detailed

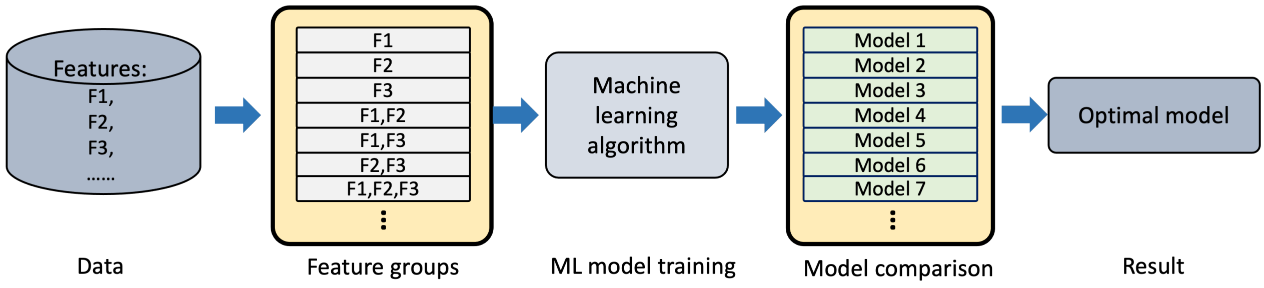


Figure 1. The pipeline of getting an optimal ML model for a data set with multiple features.

implementations of RadiaNet. The extended version also includes an extensive evaluation of the proposed visualisation approach with user studies for additional insights.

2. Background and Related Work

In machine learning, given a fixed number of features, it is possible to use different features and their groups to train machine learning algorithms resulting in various machine learning models. Users need to compare these models to find the optimal one for their tasks. Getting the optimal results out of machine learning models requires a truly understanding of all models. However, each data set with a large number of features can have hundreds or even thousands of ML models, making it nearly impossible to understand all models based on different feature groups in an intuitive fashion. Visualisation can be used to help unlock nuances and insights in ML models.

This section investigates various visualisations from the perspectives of multi-attribute data visualisation, visualisation in explanation of machine learning, and comparison visualisation in order to demonstrate the state-of-art approaches and challenges for comparison of machine learning models with visualisation.

2.1. Visualisation of Multi-Attribute Data

The comparison visualisation of machine learning models is related to multi-attribute (or multiple features) data visualisation. The visualisation of multi-attribute data has been frequently investigated for years [11]. For example, multidimensional projections are one of effective methods for visualizing high-dimensional datasets to find structures in the data like groups of similar points and outliers. One of classical approaches to visualise multi-attribute data points is parallel coordinates [12]. The advantage of this technique is that it can provide an overview of data trend. One of obvious disadvantages of parallel coordinates is that it lacks a tabular view for presenting value details of each coordinates. SimulSort [13] organizes different attributes of data in a tabular and sorts all of the attribute columns simultaneously. However, users still need laborious interactions in SimulSort in order to highlight different points for comparison. Zhou et al. [14] proposed a visualisation approach for presenting multi-attribute data by combining advantages of both parallel coordinates and SimulSort, which organizes various attributes in a tabular-like form implicitly. Colours are used to encode data belonging to different groups, instead of highlighting attributes of one point at a time as in SimulSort. Such colour encoding approach provides an overview of points and their associated attribute details to improve the information browsing efficiency. Motivated by such colour encoding, this paper uses colours to encode ML model performance to provide an overview of performance for comparison. However, such visualisation cannot reveal complex relations between machine learning models and their dependent features with dynamic numbers.

Moreover, the contradiction between the limited space and the large amount of information to be presented is another challenge for multi-attribute data visualisation. Coordinated & multiple views (CMV) [15] is widely used to extend the limited space of a single view for large data set visualisation. Langner et al. [16] presented a framework that uses a set of mobile devices to distribute and coordinate multiple visualisation views for the exploration of multivariate data. Koytek et al. [17] proposed *MyBrush* for extending brushing and linking technique by incorporating personal agency in the interactive exploration of data relations in CMV. Sarikaya et al. [18] introduced a framework to help determine the design appropriateness of scatterplot for task support to modify/expand the traditional scatterplots to scale as the complexity and amount of data increases. Most of these investigations focus on the extension of spaces for the complex information presentation, however ignore making full use of a given limited space. Our approach in this paper aims to encode complex information with less visual elements (e.g. model lines) to avoid entangled visual elements in the limited space to improve the information presentation efficiency.

2.2. Visualisation in Explanation of Machine Learning

Yuan et al. [19] reviewed techniques of visual analytics for machine learning by categorising them into techniques before model building, techniques during modeling building, and techniques after model building. Chatzimparmpas et al. [20] investigated approaches of enhancing trust in ML models with the use of interactive visualization. Visualisation is also used in ML explanations. Corresponding to the term of Exploratory Data Analysis (EDA) in terms of the desired outcome of the analytic process, Cashman et al. [21] presented a concept of Exploratory Model Analysis (EMA) with a user-based visual analytics workflow, which is defined as the process of discovering and selecting relevant models that can be used to make predictions on a data source. However, it does not consider the comparison of models with different number of features.

In the early years, visualisation played the role to explain the learning process of simple machine learning algorithms in order to understand how the data is processed and results are got in machine learning. For example, various visualisation approaches are used to examine specific values and probabilities of picked objects visually for Naïve-Bayes [2], decision trees [22], Support Vector Machines (SVMs) [23]. Advanced visualisation techniques are then proposed to present more complex ML processes. Erra et al. [24] introduced a visual clustering which utilises a collective behavioral model, where visualisation helps users to understand and guide the clustering process. Paiva et al. [25] presented an approach that employs the similarity tree visualisation to distinguish groups of interest within the data set. Visualisation is also used as an interaction interface for users in machine learning. For example, Guo et al. [26] introduced a visual interface named Nugget Browser allowing users to interactively submit subgroup mining queries for discovering interesting patterns dynamically. EnsembleMatrix allows users to visually ensemble multiple classifiers together and provides a summary visualisation of results of these multiple classifiers [3]. Zhou et al. [27] revealed states of key internal variables of ML models with interactive visualisation to let users perceive what is going on inside a model.

More recent work tries to use visualisation as an interactive tool to facilitate ML diagnosis. ModelTracker [28] provides an intuitive visualisation interface for ML performance analysis and debugging. Chen et al. [29] proposed an interactive visualisation tool by combining ten state-of-the-art visualisation methods in ML (shaded confusion matrix, ManiMatrix, learning curve, learning curve of multiple models, McNemar Test matrix, EnsembleMatrix, Customized SmartStripes, Customized ModelTracker, confusion matrix with sub-categories, force-directed graph) to help users interactively carry out a multi-step diagnosis for ML models. Wongsuphasawat et al. [30] presented an approach called TensorFlow Graph Visualizer to visualise graphs of data flow in deep learning to help users debug, understand, and share the structure of their deep learning models.

Visualisations comprise the major body of ML process explanations. However, these approaches cannot be directly used for the comparison of machine learning models trained with a different number of features, and facilitate the revealing of feature importance directly from visualisations of models for ML explanations.

2.3. Comparison Visualisation

Supporting comparison is a common challenge in visualisation. Gleicher [7] categorized four considerations that abstract comparison when using visualisation. These four considerations include to identify: the comparative elements, the comparative challenges, a comparative strategy, and a comparative design, which provide a guideline for developing comparison solutions in visualisation. Law et al. [8] presented Duet, a visual analysis system to conduct pairwise comparisons. Duet employs minimal specification in comparison by only recommending similar and different attributes between them when one object group to be compared is specified. Qi et al. [31] presented a visual technique called STBins for visual tracking of individual data sequences and also for comparison of multiple sequences. The comparison of sequences is done by showing the similarity of sequences within temporal windows. The analysis of subtle deviations between different versions of historical prints is important but also a challenge in art history research. Plüger et al. [32] developed an approach called VeCHart that detects similar stroke-patterns in prints and matches them in order to allow visual alignment and automated deviation highlighting for comparison purposes. Cutura et al. [33] proposed a visual analysis approach called Compadre for comparing distances of high-dimensional data and their low-dimensional projections. The key of the visual analysis is a matrix visualization to represent the discrepancy between distance matrices which are linked with 2D scatter plot projections of the data. Heimerl et al. [34] introduced an interactive visualisation approach of embComp for comparing two embeddings that capture the similarity between objects, such as word and document embeddings. The proposed approach features overview visualizations that are based on metrics for measuring differences in the local structure around objects, and detail views allowing comparison of the local structure around selected objects and relating this local information to the global views. However, little work is done on the comparison of machine learning with different number of features.

Bar chart is one of commonly used visualisation methods for comparison in machine learning [6]. It works with two variables – one is the length of the bar on one axis and the second is the position of this bar on another axis. The variable is compared by denoting it with the length of the bars when various bars are plotted together. Radar Chart is another commonly used approach to compare multiple quantitative variables. It is useful for seeing which variables have similar values or if there are any outliers amongst the values of each variable. It can also help to find which variables are high or low. Besides, other methods such as line chart and ring chart are also used in comparison. Ondov et al. [9] made evaluations of comparison visualizations of 5 layouts: stacked small multiples, adjacent small multiples, overlaid charts, adjacent small multiples that are mirror symmetric and animated transitions. The data to be compared are encoded with the length of bars in bar charts, slop of lines in line charts, and angle of arcs in donut charts.

These previous works provide significant guidelines and advances in comparison visualisation. This paper proposes a new visualisation method for machine learning model comparison with a full consideration of four aspects as categorized in [7]. The new visualisation approach is evaluated by comparing it with other three commonly used visualisation methods (bar chart, line chart, and radar chart) in machine learning model comparisons.

3. RadialNet Chart

This section presents a novel visualisation approach called *RadialNet Chart* to compare machine learning models trained with different feature groups of a data set.

3.1. Design Goals

After having a thorough survey with experienced researchers and developers in machine learning on their problems meeting in comparing machine learning models, we phrase following design goals for the RadialNet:

- **Comparison**: To maximise differences among visual elements of models to help users find the optimal target easily. The comparison is the core objective in the ML model visualisation. This is a challenge when substantial amounts of ML models must be compared.
- **Importance**: To easily identify importance of features directly from visualisation. The importance of features plays significant roles in the feature selection in the ML pipeline and ML explanations [35]. It is a challenge to identify importance of features directly from visualisation without complex feature importance calculations.
- **Feature identification**: To easily identify relationships between models (and model performance) and their dependent features. This helps users easily link ML models and their

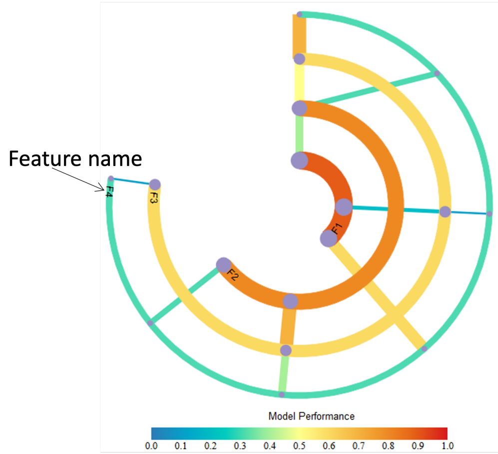


Figure 2. An example of RadialNet chart.

dependent features for understanding both features and models, which is usually challenging with commonly used visualisation approaches.

- **Compactness**: To represent complex visualisation in a compact form and reduce the visual clutters because of substantial amounts of information in a limited space.

3.2. Definition of RadialNet Chart

This subsection defines the RadialNet. Fig. 2 shows an example of RadialNet. Based on this example, we firstly give following definitions that are used to set up a RadialNet:

Feature arc Each feature is represented by a concentric arc in RadialNet. The arc is also called feature arc. The name of each feature is displayed at one end of the arc as shown in Fig. 2 (e.g. F1, F2, F3, F4). Each arc also represents the ML model based on that single feature.

Model line RadialNet uses a line segment to represent an ML model based on multiple features. The line is also called model line. For example, in Fig. 2, the line *AB*, *BC*, and *CD* represent different ML models respectively. The features used for the model are defined based on the feature path of the line (see the definition of feature path below).

Feature point A feature point refers to an intersection point of a model line with an arc. It is represented by a dot point on a feature arc as shown in Fig. 2 (e.g. feature points A, B, C).

Feature path A feature path defines features used for a model line. A feature path starts from the feature point of a model line on its outermost arc and ends at the feature point on the innermost arc it can reach through the connected feature point in the RadialNet. For example, in Fig. 2, for the model line *AB*, its feature path starts from the feature point A on the arc F4, passes through B and C, and ends at D on the innermost arc F1. This path can be represented by a list of features corresponding to arcs of each feature point, i.e. the feature path of *AB* is [F4, F3, F2, F1]. Similarly, the feature path of *BC* is [F3, F2, F1], the feature path of *CD* is [F2, F1], the feature path of *EC* is [F4, F2, F1], the feature path of *MP* is [F4, F3, F2], and the feature path of *PQ* is [F3, F2].

Furthermore, the model performance is encoded using two methods: the width of the line/arc and the colour of the line/arc. The wider the line/arc is, the higher the model performance. A colour scale is accompanied with the RadialNet to encode model performance and let users easily perceive the difference of performance of different models as shown in Fig. 2.

Based on these definitions, the visualisation of lines and arcs are spiraling from the centre to outside and therefore it is called *RadialNet Chart*. The RadialNet has different advantages. For example, given a data set in machine learning, if most of ML models related to one specific feature show high model performance, that feature can be considered as a high important feature, and vice versa if most of ML models related to one specific feature show low model performance, that feature can be considered as a less important feature. The RadialNet can depict importance of features directly through visualisation: if an arc and its connected lines are mostly wider than others and have colours representing high performance values in the colour scale, the feature represented by the arc is an important feature, and vice versa it can also depict less important features. For example, in Fig. 2, the feature F1 is an important feature because the width and colour of the arc as well as its connected lines are mostly wider and red, while the feature F4 is an less important feature. The RadialNet also helps users directly identify features used for a specific model because of the feature path mechanism in RadialNet.

Fig. 3 shows the steps used to draw a RadialNet. The definition of different parameters is the key during RadialNet drawing. Firstly, key parameters are defined with user interactions or predefined approaches. Arc parameters and line parameters are then generated based on key parameters. The RadialNet is drawn finally based on generated parameters.

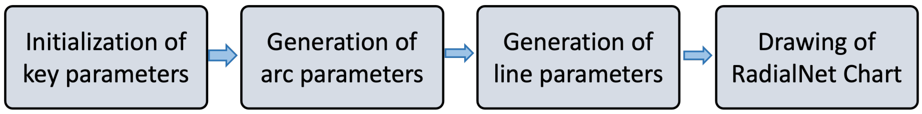


Figure 3. The steps for drawing RadialNet.

3.3. Key Parameter Initialization

The key parameters include the overall spanning angle of RadialNet, the overall number of models given the number of features, the size of the drawing canvas, as well as others. The overall spanning angle defines the space that the RadialNet covers in degrees. It can be interactively modulated by users to control the compactness of the visualisation in a limited space. If the number of ML models to be visualized is low, a small value can be defined for the spanning angle, and vice versa a large value can be defined for the spanning angle in order to help users easily control and compare ML models in a limited space.

Given N features of a data set, F1, F2, ..., FN, a machine learning algorithm uses these features to set up ML models. The ML models can be set up based on one or multiple features of the data set. Typically, the number of models based on various groups of N features can be got from Equ. 1:

$$C_N = C_N^1 + C_N^2 + ... + C_N^i + ... + C_N^N = 2^N - 1 \tag{1}$$

where C_N is the number of models based on groups of N features, C_N^i is the group number of selecting i features from N features. It shows that the number of ML models is increased exponentially with the increase of number of features.

Furthermore, because of the circular characteristics of RadialNet, polar coordinates are used to represent arcs and lines in RadialNet.

3.4. Arc Parameter Generations

Algorithm 1 shows the process for generating arc parameters. The arc is denoted by its start point and end point in polar coordinates. In this algorithm, `arcSpanning` defines the largest angle that arcs cover in the space and can be interactively changed by a sliding bar in the user interface. `N` is the number of features. `canvasWidth` is the width of the drawing canvas. `allFeatures` is a list of all studied features which are sorted in the decreased order based on model performance of individual features. Each arc represents the model performance based on an individual feature from `allFeatures` list. The algorithm generates arc parameters aiming to make N arcs evenly distributed in the drawing canvas space. This algorithm initialises the spanning angle of each arc with the `arcSpanning` value, and the spanning of each arc (`arcAngle`) is dynamically updated in the drawing algorithm (see Algorithm 3) to allow arcs in a spiral format. `arcParasDict` is a dictionary storing parameters of arcs and the key of the dictionary is the individual features for the arc. The parameters include arc's radius, spanning angle and arc width. `Data` is read from a JSON file and stores different feature groups and their model performance values.

Algorithm 1: Algorithm for arc parameter generations

```
Function ArcParasGen( arcSpanning, N, canvasWidth, allFeatures,
 Data):
    // Distance between two arcs
    arcSpacing ← canvasWidth/(2*N);
    prev_radius ← 0.0;
    arcParasDict ← { };
    for f in allFeatures do
        arcRadius ← prev_radius + arcSpacing;
        prev_radius ← arcRadius;
        // Encode performance of the model based on f as the arc
           width
        arcWidth ← Data[f].performance;
        arcAngle ← arcSpanning;
        arcParasDict[f] ← [arcRadius, arcAngle, arcWidth];
    return arcParasDict
```

3.5. Line Parameter Generations

Algorithm 2 shows a recursive function used for generating model line parameters. The line is denoted by its start point and end point in polar coordinates. In this algorithm, *lineParasDict* is a dictionary and stores parameters of lines, and the key of the dictionary is the feature list (feature path) used for the line. The line parameters stored in the dictionary include the start and end points of the line in polar coordinates as well as line width of the line. *lineFeatures* is the feature list for the current line and is sorted in the decreased order based on model performance of individual features. *startAngle* is the angle of polar coordinates of the start point of the line. *angleStep* is the step size that angle increases each time.

In this algorithm, if the key with the current *lineFeatures* does not exist in *lineParasDict*, a sub-key with the feature list by removing the last feature in *lineFeatures* is created. If this sub-key still does not exit in *lineParasDict* and the number of features in this sub-key is more than 2, the algorithm recursively call this function with the current sub-key features. Otherwise, the algorithm defines the start point and end point of the line and pushes them into *lineParasDict*.

The line width is encoded with the model performance based on *lineFeatures*. The colour of the line is also encoded with the model performance using a colour scale.

3.6. RadialNet Chart Drawing

Algorithm 3 shows the process of drawing a RadialNet. In Algorithm 3, after getting key parameters such as number of points on the outermost arc and arc spanning angle, Algorithm 1 is firstly called to generate arc parameters. Then Algorithm 2 is called for each feature to generate line parameters related to that feature. These parameters are then used to draw arcs and lines by calling functions of DrawArcs() and DrawLines() respectively. DrawArcs() and DrawLines() calls Javascript functions to draw arcs and lines.

4. Implementation

The proposed approach is implemented in Javascript based on the D3.js library [36]. The data input to RadialNet are saved in a JSON file. The RadialNet is also implemented as a Javascript library and it is easily to be reused in different visualisation applications. This library will be released as an open-source library.

5. Case Studies

In this section, RadialNet is used to visualise machine learning models based on different data sets and ML algorithms. Two data sets from UCI machine learning data repository [37] and PPMI [38] respectively were analyzed, and three machine learning algorithms of K-Nearest

Algorithm 2: Algorithm for line parameter generations

```
Function LineParasGen(allFeatures, lineParasDict, arcParasDict,
 lineFeatures, startAngle, angleStep, Data):
    // Use lineFeatures as key of lineParasDict
    ikey ← lineFeatures;
    len_lineFeatures ← lineFeatures.length;
    if ikey is not in lineParasDict then
        // Sub-features without the last feature
        isubkey ← ikey[:len_lineFeatures-1];
        len_isubkey ← isubkey.length;
        if isubkey is not in lineParasDict and len_isubkey> 2 then
            // Recursively call the function
            LineParasGen(allFeatures, lineParasDict, arcParasDict,
             isubkey, startAngle, angleStep, Data);
        else
            // Define start and end points
            if isubkey is in lineParasDict then
                // Polar coordinates of start point
                startAngle←lineParasDict[isubkey].endAngle;
                startRadius←lineParasDict[isubkey].endRadius;
                endSubF ← isubkey.endFeature;
                endF ← ikey.endFeature;
                if not neighbour(endSubF, endF) in allFeatures then
                    dist ← distance(endF, endSubF) in allFeatures;
                    startAngle←startAngle+angleStep*dist;
            else
                if lineFeatures.length == 2 then
                    startAngle←startAngle+angleStep;
                iFeature ← lineFeatures[len_lineFeatures-1];
                startRadius ← arcParasDict[iFeature].radius;
            // Polar coordinates of end point
            lastFeature ← lineFeatures[len_lineFeatures];
            endAngle ← startAngle;
            endRadius ← arcParasDict[lastFeature];
            // Encode model performance as the line width
            lineWidth ← Data[lineFeatures].performance;
            // Push line parameters into dict
            lineParasDict[ikey] ← [startAngle, startRadius, endAngle,
             endRadius, lineWidth];
    return lineParasDict, startAngle
```

Neighbours (KNN), Naïve Bayes (NB) and Random Forest (RF) were deployed in the experiment. Fig. 4 shows the visualisation of different ML models for a data set with 6 features. From this visualisation, we can easily locate the model with the highest performance (the widest red line *AB* as shown in Fig. 4) as well as features (two features of "alcohol" and "pH" on the feature path of the line) used for the model training. It also helps users easily identify the importance of features, the most important feature "alcohol" is represented by the outermost arc (the arc and its connected lines are mostly redder and wider than others) and the lest important feature "free suffur" is represented by the innermost arc (the arc and its connected lines are mostly bluer and narrower than others). Fig. 5 shows the visualisation of different ML models for a data set with 7 features. Compared with Fig. 4, the model number is increased dramatically when the feature number is increased just one. This visualisation also helps users easily locate the model with the lowest performance (the narrowest blue line *AB* as shown in Fig. 5). We can also easily directly

Algorithm 3: Algorithm for drawing RadialNet

```
Input: allFeatures, arcSpanning, N, canvasWidth, Data
Output: SpiralChart
// Number of points on the outmost arc
num_points ← C_{N−1}; // see Equ. 1;
// Define step size of angles
angleStep ← 2*arcSpanning / (num_points - 1);
// Initialize parameters
startAngle ← 0;
lineParasDict ← { };
// Generate arc parameters
arcParasDict ← ARCPARASGEN(arcSpanning, N, canvasWidth);
// Generate line parameters
for f in allFeatures do
    // Number of lines based on feature f
    num_lines ← Data[f].length;
    for j ← 1 to num_lines do
        // Feature list used for the current line
        lineFeatures ← Data[f][j];
        // Number of features for the current line
        num_features ← lineFeatures.length;
        if num_features != 1 then
            // Generate line parameters
            lineParasDict, startAngle ← LINEPARASGEN (allFeatures,
             lineParasDict, arcParasDict, lineFeatures, startAngle,
             angleStep);
            // Update arcAngle
            if j == num_lines then
                arcParasDict[f].arcAngle ← startAngle/2;
// DrawLines and DrawArcs call Javascript functions to draw lines
   and arcs of RadialNet Chart
DrawArcs (arcParasDict);
DrawLines (lineParasDict);
```

identify the most important feature (the third inner arc represented by the widest red arc) and the least important feature (the innermost narrowest yellow arc) as shown in Fig. 5.

Besides comparison of feature importance of a data in RadialNet, it can also be used to compare performance of different ML algorithms for a given data set. Fig. 6 shows the comparison of three ML algorithms for the same data set with RadialNet visualisation. From this figure, we can easily get that the ML algorithm represented by the left diagram shows the worst performance, compared to algorithms represented by the other two diagrams, because its colour is bluer which is located on the left side of the colour scale. While the algorithm represented by the middle diagram shows the best performance because its colour is redder which is located on the right side of the colour scale. Furthermore, the visualisation shows that the feature represented by the outermost arc (i.e. the feature of "alcohol") is the most important feature because this arc is the widest and its colour is located on the right side of the colour scale in all three visualizations.

6. Evaluation

To understand the effectiveness of RadialNet in the ML model comparison, we compare it with three commonly used visualisation approaches of bar chart, line chart and radar chart. 11 participants were recruited (9 males and 2 females, ages from 20s-40s) to conduct a comparison

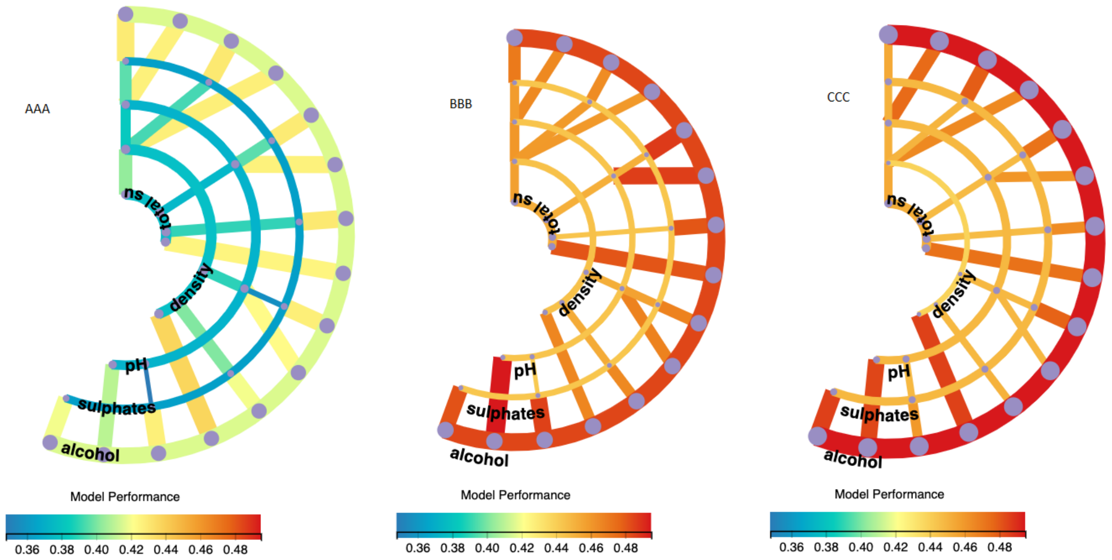


Figure 6. Comparison of three ML algorithms for the same data set with RadialNet.

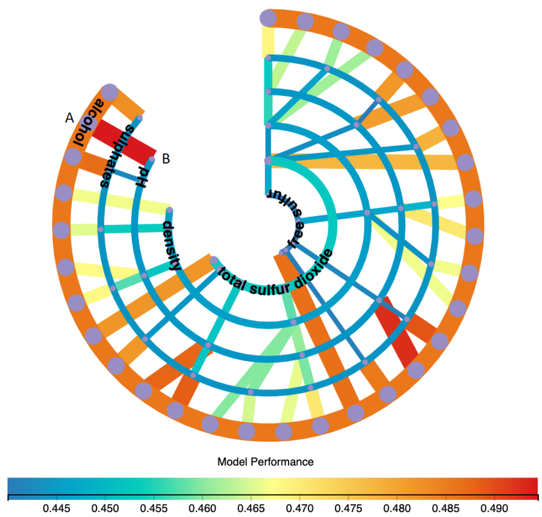


Figure 4. RadialNet of ML models based on a data set with 6 features.

user study. All participants are researchers and developers experienced in machine learning related areas.

The following metrics are proposed to evaluate different visualisations:

- **Comparison**: How easily that the visualisation helps users to compare performance of different models;
- **Feature importance**: How easily that the visualisation helps users to identify importance of features;
- **Feature identification**: How easily that the visualisation helps users to link each model and its dependent features;
- **Complexity**: How complex the visualisation is to present data.

Besides, user cognitive responses to visualisation such as mental effort as well as time spent on the selection task are also evaluated to compare effectiveness of visualizations:

- **Mental effort**: How much mental effort users used for tasks with the visualisation;
- **Time spent**: How much time users spent in task decisions with the visualisation.

To understand the usability of the RadialNet Chart, we also administrate a questionnaire that asks participants questions about their experience and feedback in using the charts. Further, eye

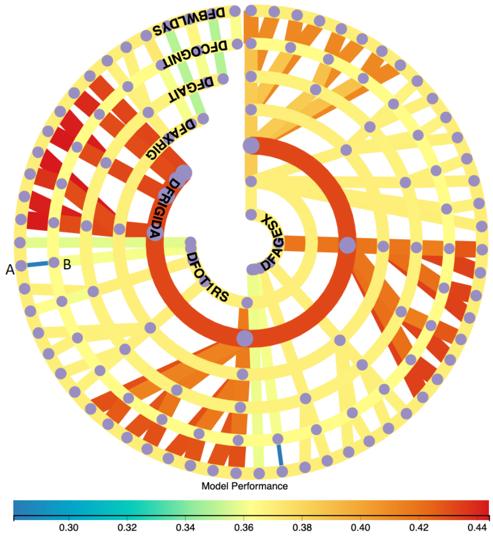


Figure 5. RadialNet of ML models based on a data set with 7 features.

tracking study is conducted with a separate participant to understand participant's eye movement behaviours with different visualisations.

6.1. Data and Visualisation

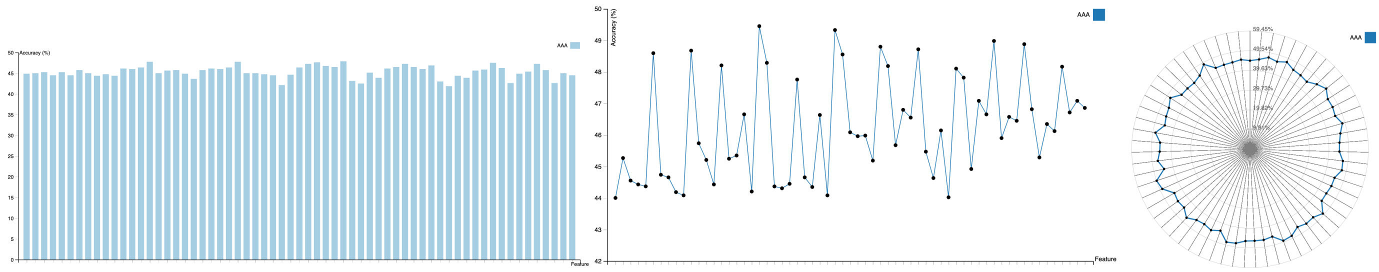
Two data sets from UCI machine learning data repository [37] and PPMI [38] respectively were analysed in this study. Two data sets have 6 features and 7 features respectively, which generate 63 ML models and 127 ML models respectively to compare. ML models are visualised using bar chart, line chart, radar chart, and RadialNet respectively as shown in Fig. 7 and Fig. 4 (the data set with 6 features visualised in Fig. 7 and Fig. 4). In bar chart, line chart, radar chart and RadialNet, the related features for a model and its performance are popped up when the mouse is moved over the relevant visual elements (e.g. bars, dots, lines, or arcs), which allows users to inspect more details of each model.

Besides, for a given data set, three ML algorithms were used generating various ML models respectively. The ML models by these three ML algorithms were visualised together in a single bar chart, line chart, and radar chart respectively as shown in Fig. 8, which were also visualised using RadialNet as shown in Fig. 6. These visualisations were used to compare the effectiveness of different ML algorithms. AAA, BBB, and CCC in visualisations (e.g. Fig. 7, Fig. 8) represent three ML algorithms used to compare: KNN, NB, and RF. The exact ML algorithms used for ML models were not shown to participants during the study to avoid any bias.

6.2. Procedure and Data Collection

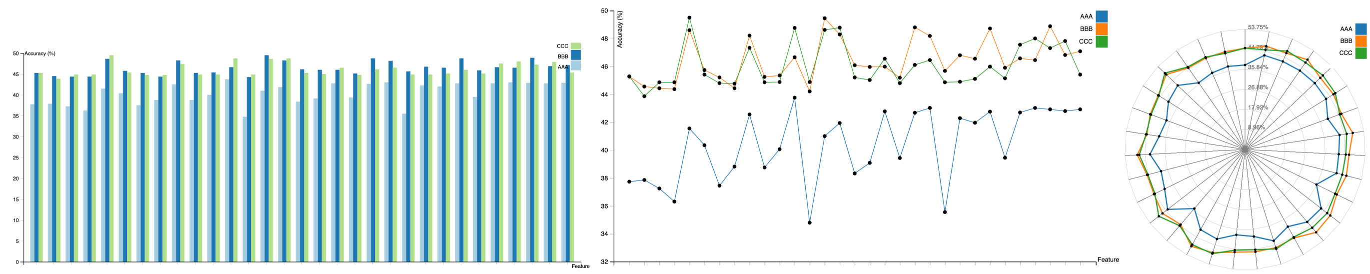
The study was conducted in a lab environment using a Macbook Pro with 13-inch display of resolution 2560×1600. The procedure of the study is described as follows: Tutorial slides on the study were firstly presented to participants to let them understand concepts and operations during the study. A training task was then conducted to practice interactions. After that, the formal tasks were conducted with different visualisations. During the study, different visualisations as described in the previous section were displayed to participants one-by-one in random order. For each visualisation, participants were firstly required to find which ML model gives the best or worst performance by selecting the visual elements in the visualisation (we call it as the selection task). This is more akin to what analysts do with real data sets. After the selection task, participants were asked to answer different questions as described below on the task and visualisation. At the end of the study, participants were asked to give their feedback in using the charts and some personal details such as gender, age, working topics.

After the selection task of each visualisation, the participants were asked to answer questions related to comparison, feature importance, feature identification, visual complexity, and mental effort on the visualisation using 9-point Likert scales (comparison, feature importance, feature



(a) Bar chart. **(b)** Line chart. **(c)** Radar chart.

Figure 7. ML models based on the data set of 6 features are visualized using bar chart, line chart and radar chart respectively.



(a) Bar chart. **(b)** Line chart. **(c)** Radar chart.

Figure 8. Comparison of three ML algorithms for the same data set with three visualisation approaches.

identification: 1=least easiness, 9=most easiness; visual complexity: 1=least complex, 9=most complex; mental effort: 1=least effort, 9=most effort). At the end of all visualisation tasks, the participants were also asked to answer in a questionnaire which visualisation helps users more easily compare ML performance of different features, and which visualisation helps users more easily compare ML performance of different ML algorithms respectively.

6.3. Results

In this section, for the evaluation of each metrics, we firstly performed one-way ANOVA test and then followed it up with post-hoc analysis using t-tests (with a Bonferroni correction under a significance level set at $p < \frac{.05}{4} = .013$, based on the fact that we had four visualisation types to test) to analyze differences in participant responses of each metrics. Each metric values were normalised with respect to each subject to minimise individual differences in rating behavior (see Equ. 2):

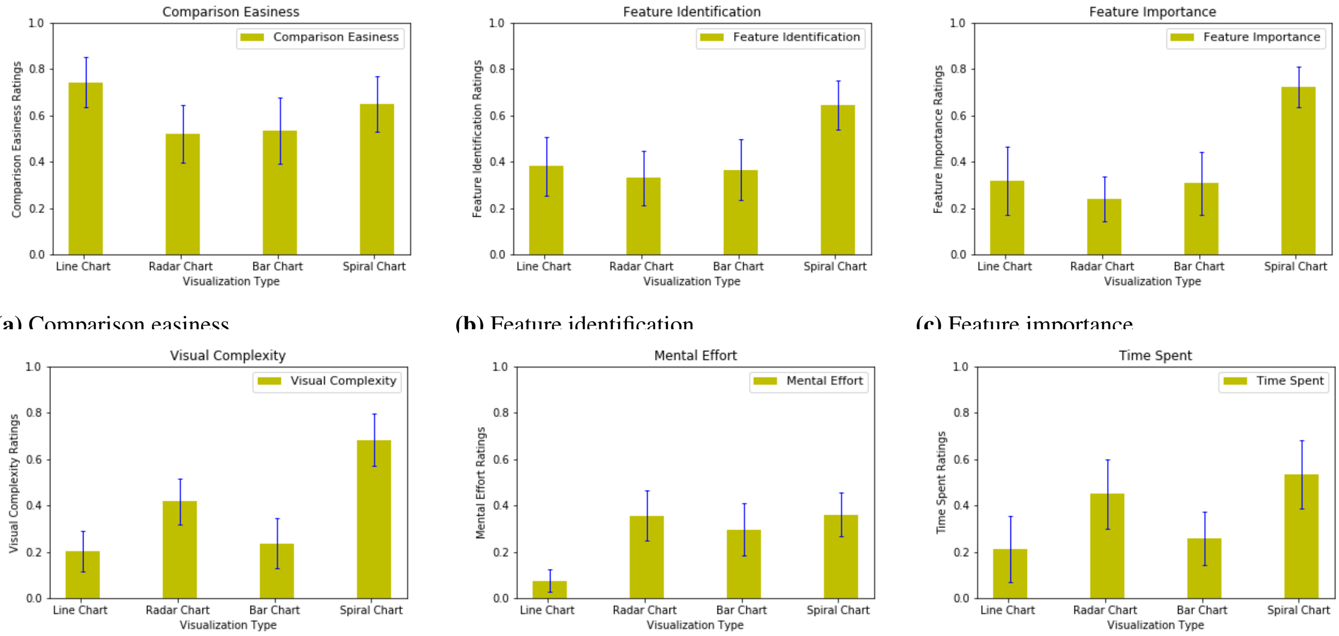
$$T_i^N = \frac{T_i - T_i^{min}}{T_i^{max} - T_i^{min}} \tag{2}$$

where T_i and T_i^N are the original metric rating and the normalised metric rating respectively from the participant i, T_i^{min} and T_i^{max} are the minimum and maximum of metric ratings respectively from the participant i in all of his/her tasks. The time spent in the selection tasks is also normalised in a similar way as other five metrics.

Fig. 9 shows mean normalised metric values for different visualisation types.

Comparison easiness One-way ANOVA test gave significant differences in comparison easiness among four visualisation types ($F(3,84) = 3.067$, $p < .03$) (see Fig. 9a). However, the post-hoc t-tests only found that line chart was significantly easier to compare performance of different ML models than radar chart ($t = 2.813, p < .007$). The result shows that RadialNet did not help users increase the easiness in comparing performance of different ML models, which is not as we expected, but a trend shows the higher ratings in comparison easiness for RadialNet than bar chart and radar chart (see Fig. 9a). This is maybe because of the relatively small number of participants used for the study.

Feature identification One-way ANOVA test found significant differences in easiness of feature identification among four visualisation types ($F(3,84) = 6.108$, $p < .001$) (see Fig. 9b).



(a) Comparison easiness. **(b)** Feature identification. **(c)** Feature importance.

(d) Visual complexity. **(e)** Mental effort. **(f)** Time spent.

Figure 9. Comparison of mean normalized metrics for different visualisation types.

The post-hoc t-tests found that RadialNet was significantly easier to identify features related to models than all of other three visualisation types (line chart: $t = 3.296, p < .002$; bar chart: $t = 3.393, p < .002$; radar chart: $t = 4.089, p = .000$). This is because that users can get features and performance related to an ML model directly from connected visual elements in RadialNet, while users need to move mouses to visual elements of each model to inspect related features and performance in other three visualisations.

Feature importance There were significant differences found in easiness of identifying feature importance among four visualisation types by one-way ANOVA test ($F(3, 84) = 14.481$, $p = .000$) (see Fig. 9c). The post-hoc t-tests found that RadialNet was significantly easier to identify feature importance than all of other three visualisation types (line chart: $t = 4.878, p = .000$; bar chart: $t = 5.320, p = .000$; radar chart: $t = 7.678, p = .000$). The results suggest the obvious advantage of RadialNet over other three visualisation types for feature importance identifications.

Visual complexity One-way ANOVA test found significant differences in visual complexity among four visualisation types ($F(3, 84) = 20.254$, $p = .000$) (see Fig. 9d). The post-hoc t-tests found that RadialNet was significantly more complex than all of other three visualisation types (line chart: $t = 7.032, p = .000$; bar chart: $t = 6.001, p = .000$; radar chart: $t = 3.710, p < .001$). It was also found that radar chart was significantly more complex than line chart ($t = 3.383, p < .002$).

Mental effort There were significant differences found in mental effort among four visualisation types by one-way ANOVA test ($F(3, 84) = 8.757$, $p = .000$) (see Fig. 9e). The post-hoc tests found that line chart took significantly less effort than other three visualisation types (bar chart: $t = 3.722, p < .001$; radar chart: $t = 4.981, p = .000$; RadialNet: $t = 5.562, p = .000$). RadialNet did not show significant differences in mental effort with radar chart and bar chart.

Time spent One-way ANOVA test found significant differences in time spent in the selection of the best/worst model task among four visualisation types ($F(3, 84) = 5.301$, $p < .002$) (see Fig. 9f). The post-hoc tests found that users spent significantly more time in RadialNet than in both line chart ($t = 3.286, p < .002$) and bar chart ($t = 3.111, p < .003$) respectively.

When four types of visualisation were used to compare performance of different ML algorithms for a given data set, it was found that line chart was easier to compare performance of different ML algorithms followed by RadialNet despite no significant differences found in the easiness. This could be because of the relatively small number of participants in this study. However, RadialNet can reveal importance of features while others not when comparing performance of different ML algorithms.

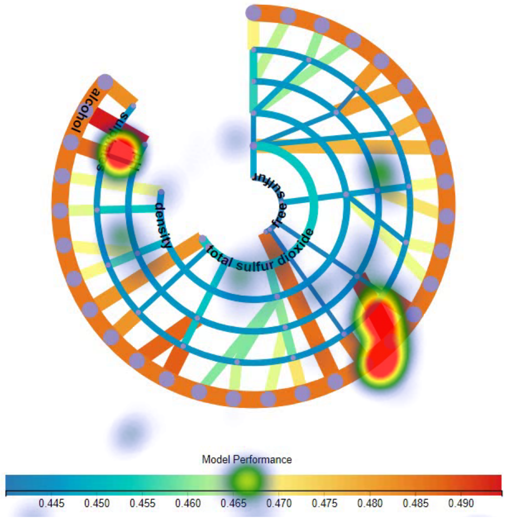


Figure 11. Heat map of RadialNet.

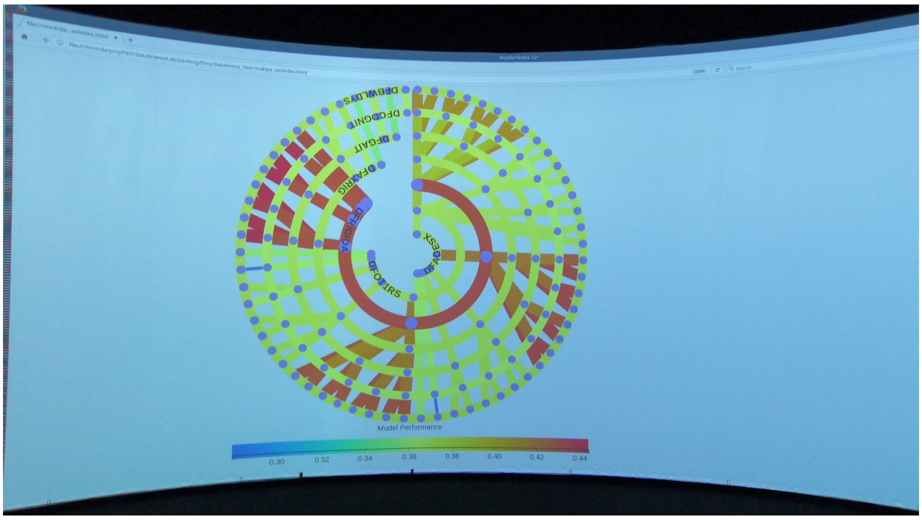
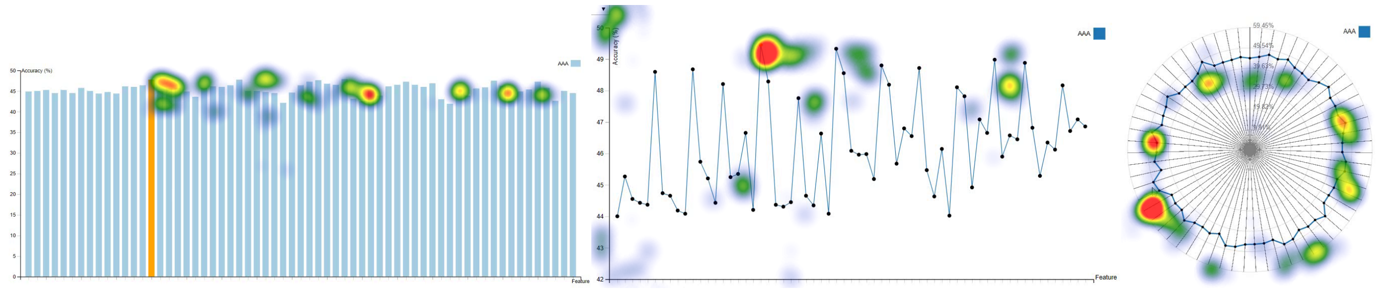


Figure 12. RadialNet displayed in our large scale visualisation facility.

We also collected participants' feedback after completing all tasks by each participant. Overall, all participants believed that "RadialNet is the most effective visualisation in identifying feature importance compared with other three approaches". Some participants suggested to "enlarge the size of RadialNet with the increase of number of features". Participants agreed that "RadialNet is more efficient to help users focus their attention to find visual elements of interest". Fig. 10 and Fig. 11 show heat maps on four visualisations recorded by an SMI eye-tracker from a participant during the selection task period respectively. Heat maps reveal the focus of attention by colours indicating the amount of time eyes stay focused on a particular area in the visualisation, the redder, the more time eyes focused. Fig. 10 and Fig. 11 suggest that the user's attention in RadialNet was more focused on two model lines with high performance (wide red lines), while it was much scattered among different points in other three visualisations.

Overall, we can say that RadialNet shows significant advantages in identifying features and performance related to specific models as well as easily revealing importance of features



(a) Bar chart **(b)** Line chart **(c)** Radar chart

Figure 10. Heat maps of bar chart, line chart, and radar chart.

compared with other three visualisation types. Despite these advantages, the mental effort and time spent in RadialNet did not show much differences from others such as radar chart.

7. Discussion

This study proposed a novel visualisation approach to compare variables with different number of dependents. Data information is encoded with colour, line width, as well as structure of visualisation to reveal insights from data. The experimental results showed that RadialNet has advantages in identifying features related to specific models as well as directly revealing importance of features for ML explanations. Different from conventional feature importance evaluations based on complex computing algorithms (such as by simulating lack of knowledge about the values of the feature(s) [39], or by mean decrease impurity, which is defined as the total decrease in node impurity averaged over all trees of the ensemble in Random Forest[40]), RadialNet allows users to estimate feature importance directly from visualisation by checking lines connected to the feature arc. The consistent large line width of these lines with colours on the right-hand side of the colour scale indicate the high importance of the feature to the modelling.

RadialNet is more compact to show more information in a limited space compared with other three visualisation types. And the compactness of RadialNet can also be controlled by changing its spanning angle dynamically (see the attached video with this paper). However, RadialNet will be much complex when the number of features is high. This could be compensated with large scale visualisation facilities. For example, our visualisation facility provides a 360-degree interactive visualisation, which change the way we view and interact with data. This visualisation facility is a large cylindrical screen with four metres high and ten metres in diameter. Six 3D-stereo video projectors, driven by a high performance computer graphics system, project visualisations on the cylindrical screen. Picture clarity is made possible from an image that's $20,000 \times 1200$ pixels. Viewers stand in the middle of the cylinder to interact visualisations. This facility can be used to present RadialNet with large number of ML models for effective interactions. Fig. 12 shows an example of RadialNet displayed with around 60-degree field of view in the facility.

This paper used the exploration of performance of ML models based on different feature groups from a given data set as a case study to demonstrate the powerfulness of RadialNet in visualising data with complex relations. The RadialNet can also be generalised to other applications where similar relations need to be explored.

8. Conclusion

This paper presented *RadialNet Chart*, a novel visualisation approach to compare ML models with different number of features while revealing implicit dependent relations. The RadialNet is developed to address the challenges faced in comparing a large amount of ML models with each dependent on a dynamic number of features. It is implemented by representing ML models and features with lines and arcs respectively, which in turn are generated by a recursive function and a feature path concept. We presented our design criteria and described the algorithms for generating the chart. Two case studies were also presented with representative data sets and an experiment was conducted evaluating the effectiveness of the RadialNet. Our case studies showed that the proposed visualisation can help users easily locate target models and important features. Furthermore, the user study revealed that in comparison with other commonly used visualisation approaches, RadialNet is more efficient to help users focus their attention to find visual elements of interest. It is also more compact to show more information in a limited space. Our research provides an effective visualisation approach to represent data with complex relations. It is specifically helpful for users to find optimal machine learning model and discern feature importance visually and directly, but not through complex algorithmic calculations for ML explanations.

Author Contributions: Conceptualization, J.Z. and F.C.; methodology, J.Z. and W.H.; software, J.Z. and W.H.; formal analysis, J.Z. and F.C.; investigation, J.Z.; resources, W.H.; data curation, J.Z.; writing—original draft preparation, J.Z. and W.H.; writing—review and editing, F.C.; visualization, J.Z. and W.H.; supervision, F.C; project administration, F.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: The study was approved by the Human Research Ethics Committee (HREC) of University of Technology Sydney (ETH19-3400, January 2019).

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

1. Card, S.K.; Mackinlay, J.D.; Shneiderman, B. *Readings in Information Visualization: Using Vision to Think*; Morgan Kaufmann, 1999.
2. Becker, B.; Kohavi, R.; Sommerfield, D. Visualizing the Simple Bayesian Classifier. In *Information visualization in data mining and knowledge discovery*; Fayyad, U.; Grinstein, G.G.; Wierse, A., Eds.; Morgan Kaufmann, 2001; pp. 237–249.
3. Talbot, J.; Lee, B.; Kapoor, A.; Tan, D.S. EnsembleMatrix: interactive visualization to support machine learning with multiple classifiers. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2009, pp. 1283–1292.
4. Wu, A.; Wang, Y.; Shu, X.; Moritz, D.; Cui, W.; Zhang, H.; Zhang, D.; Qu, H. Survey on Artificial Intelligence Approaches for Visualization Data. *CoRR* **2021**, *abs/2102.01330*, [2102.01330].
5. Viegas, F.; Wattenberg, M. Visualization: The secret weapon for machine learning. Keynote in EuroVis 2017, 2017.
6. Aigner, W.; Miksch, S.; Schumann, H.; Tominski, C. *Visualization of Time-Oriented Data*; Human-Computer Interaction Series, Springer, 2011.
7. Gleicher, M. Considerations for Visualizing Comparison. *IEEE Transactions on Visualization and Computer Graphics* **2018**, *24*, 413–423.
8. Law, P.; Basole, R.C.; Wu, Y. Duet: Helping Data Analysis Novices Conduct Pairwise Comparisons by Minimal Specification. *IEEE Transactions on Visualization and Computer Graphics* **2019**, *25*, 427–437.
9. Ondov, B.D.; Jardine, N.; Elmqvist, N.; Franconeri, S. Face to Face: Evaluating Visual Comparison. *IEEE Transactions on Visualization and Computer Graphics* **2019**, *25*, 861–871.
10. Zhou, J.; Huang, W.; Chen, F. A Radial Visualisation for Model Comparison and Feature Identification. IEEE PacificVis 2020, 2020, pp. 226–230.
11. Tian, Z.; Zhai, X.; van Driel, D.; van Steenpaal, G.; Espadoto, M.; Telea, A. Using multiple attribute-based explanations of multidimensional projections to explore high-dimensional data. *Computers & Graphics* **2021**, *98*, 93–104.
12. Guo, H.; Xiao, H.; Yuan, X. Scalable Multivariate Volume Visualization and Analysis Based on Dimension Projection and Parallel Coordinates. *IEEE Transactions on Visualization and Computer Graphics* **2012**, *18*, 1397–1410.
13. Kim, S.; Dong, Z.; Xian, H.; Upatising, B.; Yi, J.S. Does an Eye Tracker Tell the Truth about Visualizations?: Findings while Investigating Visualizations for Decision Making. *IEEE Trans. Vis. Comput. Graph.* **2012**, *18*, 2421–2430.
14. Zhou, J.; Sun, J.; Chen, F.; Wang, Y.; Taib, R.; Khawaji, A.; Li, Z. Measurable Decision Making with GSR and Pupillary Analysis for Intelligent User Interface. *ACM Transactions on Computer-Human Interaction* **2015**, *21*, 33.

15. Roberts, J.C. State of the Art: Coordinated & Multiple Views in Exploratory Visualization. Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV '07), 2007, pp. 61–71.
16. Langner, R.; Horak, T.; Dachselt, R. VisTiles: Coordinating and Combining Co-located Mobile Devices for Visual Data Exploration. *IEEE Transactions on Visualization and Computer Graphics* **2018**, *24*, 626–636.
17. Koytek, P.; Perin, C.; Vermeulen, J.; André, E.; Carpendale, S. MyBrush: Brushing and Linking with Personal Agency. *IEEE Transactions on Visualization and Computer Graphics* **2018**, *24*, 605–615.
18. Sarikaya, A.; Gleicher, M. Scatterplots: Tasks, Data, and Designs. *IEEE Transactions on Visualization and Computer Graphics* **2018**, *24*, 402–412.
19. Yuan, J.; Chen, C.; Yang, W.; Liu, M.; Xia, J.; Liu, S. A survey of visual analytics techniques for machine learning. *Comput. Vis. Media* **2021**, *7*, 3–36.
20. Chatzimparmpas, A.; Martins, R.M.; Jusufi, I.; Kucher, K.; Rossi, F.; Kerren, A. The State of the Art in Enhancing Trust in Machine Learning Models with the Use of Visualizations. *Comput. Graph. Forum* **2020**, *39*, 713–756.
21. Cashman, D.; Humayoun, S.R.; Heimerl, F.; Park, K.; Das, S.; Thompson, J.; Saket, B.; Mosca, A.; Stasko, J.T.; Endert, A.; Gleicher, M.; Chang, R. A User-based Visual Analytics Workflow for Exploratory Model Analysis. *Comput. Graph. Forum* **2019**, *38*, 185–199.
22. Ankerst, M.; Elsen, C.; Ester, M.; Kriegel, H.P. Visual classification: an interactive approach to decision tree construction. Proceedings of KDD '99, 1999, pp. 392–396.
23. Caragea, D.; Cook, D.; Honavar, V.G. Gaining insights into support vector machine pattern classifiers using projection-based tour methods. Proceedings of KDD '01, 2001, pp. 251–256.
24. Erra, U.; Frola, B.; Scarano, V. An Interactive Bio-inspired Approach to Clustering and Visualizing Datasets. Proceedings of the 15th International Conference on Information Visualisation 2011, 2011, pp. 440–447.
25. Paiva, J.G.; Florian, L.; Pedrini, H.; Telles, G.; Minghim, R. Improved Similarity Trees and their Application to Visual Data Classification. *IEEE Transactions on Visualization and Computer Graphics* **2011**, *17*, 2459–2468.
26. Guo, Z.; Ward, M.O.; Rundensteiner, E.A. Nugget Browser: Visual Subgroup Mining and Statistical Significance Discovery in Multivariate Datasets. Proceedings of the 15th International Conference on Information Visualisation, 2011, pp. 267–275.
27. Zhou, J.; Khawaja, M.A.; Li, Z.; Sun, J.; Wang, Y.; Chen, F. Making Machine Learning Useable by Revealing Internal States Update – A Transparent Approach. *International Journal of Computational Science and Engineering* **2016**, *13*, 378–389.
28. Amershi, S.; Chickering, M.; Drucker, S.M.; Lee, B.; Simard, P.; Suh, J. ModelTracker: Redesigning Performance Analysis Tools for Machine Learning. Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, 2015, pp. 337–346.
29. Chen, D.; Bellamy, R.K.E.; Malkin, P.K.; Erickson, T. Diagnostic visualization for non-expert machine learning practitioners: A design study. 2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 2016, pp. 87–95.
30. Wongsuphasawat, K.; Smilkov, D.; Wexler, J.; Wilson, J.; Mané, D.; Fritz, D.; Krishnan, D.; Viégas, F.B.; Wattenberg, M. Visualizing Dataflow Graphs of Deep Learning Models in TensorFlow. *IEEE Transactions on Visualization and Computer Graphics* **2018**, *24*, 1–12.
31. Qi, J.; Bloemen, V.; Wang, S.; van Wijk, J.; van de Wetering, H. STBins: Visual Tracking and Comparison of Multiple Data Sequences Using Temporal Binning. *IEEE Transactions on Visualization and Computer Graphics* **2020**, *26*, 1054–1063.
32. Pflüger, H.; Thom, D.; Schütz, A.; Bohde, D.; Ertl, T. VeCHArt: Visually Enhanced Comparison of Historic Art Using an Automated Line-Based Synchronization Technique. *IEEE Transactions on Visualization and Computer Graphics* **2020**, *26*, 3063–3076.
33. Cutura, R.; Aupetit, M.; Fekete, J.D.; Sedlmair, M. Comparing and Exploring High-Dimensional Data with Dimensionality Reduction Algorithms and Matrix Visualizations. Proceedings of the International Conference on Advanced Visual Interfaces, 2020, AVI '20.
34. Heimerl, F.; Kralj, C.; Moller, T.; Gleicher, M. embComp: Visual Interactive Comparison of Vector Embeddings. *IEEE Transactions on Visualization and Computer Graphics* **2020**.
35. Biran, O.; Cotton, C. Explanation and Justification in Machine Learning: A Survey. Proceedings of the 2017 IJCAI Explainable AI Workshop, 2017, pp. 8–13.
36. Bostock, M.; Ogievetsky, V.; Heer, J. D3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics* **2011**, *17*, 2301–2309.
37. Dua, D.; Karra Taniskidou, E. UCI Machine Learning Repository, 2017.

38. Prakash, N.; Caspell-Garcia, C.; Coffey, C.; et al.. Feasibility and safety of lumbar puncture in the Parkinson's disease research participants: Parkinson's Progression Marker Initiative (PPMI). *Parkinsonism & Related Disorders* **2019**.
39. Robnik-Sikonja, M.; Kononenko, I.; Strumbelj, E. Quality of Classification Explanations with PRBF. *Neurocomput.* **2012**, *96*, 37–46.
40. Breiman, L.; Friedman, J.; Olshen, R.; Stone, C. *Classification and Regression Trees*; Wadsworth and Brooks: Monterey, CA, 1984.