Movies Dataset

By Artin Elhamirad

Movies Dataset

This dataset is derived from Kaggle "The Movies Dataset" .I'll be working with "movies_metadata", the metadata for over 45,000 movies listed in the Full MovieLens Dataset. The dataset consists of movies released on or before July 2017. Data points include budget, revenue, release dates, languages, production companies, countries, TMDB vote counts and vote averages, etc.

Through data information, I can see that the original dataset has 45466 entries, and 24 total columns. Most of columns are object value which cannot be directly feed into neural networks and the other models.

Our goal in this modeling is to predict if a movie has a good revenue or not(**binary classification**)

Overview

* 1) Data Gathering and Description

* 2) Data Preparation

* 3) Data Wrangling

* 4) Modeling

* 4-1) Deep Learning Model #1: Dense Neural Network (without dropout)

* 4-2) Deep Learning Model #2: Dense Neural Network (with dropout)

* 4-3) Deep Learning Model #3: Dense Neural Network (with dropout but with different layers)

* 4-4) KNN Model: K nearest neighbour

* 4-5) SVM Model: Support Vector Machine

* 4-6) RF Model: Random Forest Algorithm

* 4-7) RBF NN: Radial Basis Neural Network

* Permutation Feature Importance for Neural Network

* Results

* Conclusion

Data Gathering and Description

In this part, I'll set up the environment, import modules, read the data, and explore our data. Through the data processing, I'll change data types to fit models, handle missing values, and detect any outliers. After that, I'll wrangle the data, split data into X and y, and do some scalling preparing for modeling.

Below are columns of the dataset:

* adult - Belongs to adult movies or not
* belongs_to_collection - Belong to movie collections or not
* budget - The budget of a movie. Some movies don't have this, it appears as 0
* genres - Main genre of the movie
* homepage - The website where can see the movie
* id - Identifier column
* imdb_id - Movies id on IMDB
* original_language - Original language of film
* original_title - Original title of film
* overview - Movie content overview
* popularity - shows weather it is popular
* poster_path - jpg. path of the movie poster

* production_companies - The production company

* production_countries - Country of origin

* release_date - Release date (YYYY-MM-DD)

* revenue - The revenue of a movie. Some movies don't have this, it appears as 0

* runtime - Duration of the movie

* spoken_languages - Languages spoken throughout the film

* status - Release or others

* tagline - Movie tagline to advertise

* title - English title

* video - True of False

* vote_average - TMDB vote average

* vote_count - TMDB vote count

A view on data :

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	...	release_date	revenue	runtime	spoken_languages	status
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	tt0114709	en	Toy Story	Led by Woody, Andy's toys live happily in his ...	...	1995-10-30	373554033.0	81.0	[{'iso_639_1': 'en', 'name': 'English'}]	Released
1	False	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN	8844	tt0113497	en	Jumanji	When siblings Judy and Peter discover an encha...	...	1995-12-15	262797249.0	104.0	[{'iso_639_1': 'en', 'name': 'English'}, {'iso...	Released
2	False	{'id': 119050, 'name': 'Grumpy Old Men Collect...	0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...	NaN	15602	tt0113228	en	Grumpier Old Men	A family wedding reignites the ancient feud be...	...	1995-12-22	0.0	101.0	[{'iso_639_1': 'en', 'name': 'English'}]	Released
3	False	NaN	16000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	NaN	31357	tt0114885	en	Waiting to Exhale	Cheated on, mistreated and stepped on, the wom...	...	1995-12-22	81452156.0	127.0	[{'iso_639_1': 'en', 'name': 'English'}]	Released
4	False	{'id': 96871, 'name': 'Father of the Bride Col...	0	[{'id': 35, 'name': 'Comedy'}]	NaN	11862	tt0113041	en	Father of the Bride Part II	Just when George Banks has recovered from his ...	...	1995-02-10	76578911.0	106.0	[{'iso_639_1': 'en', 'name': 'English'}]	Released

Data Prepration

Null data and cleaning dataset

Because of absence of some data in ‘revenue’ column, we have to remove the movies that we don’t have any data about it’s revenue.

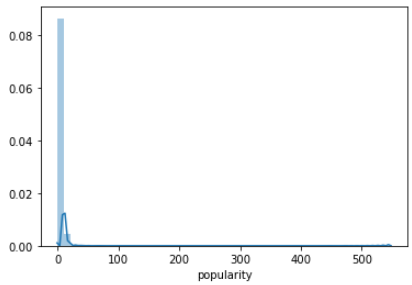
I see that the majority of the movies have a recorded revenue of 0. This indicates that I do not have information about the total revenue for these movies. Although this forms the majority of the movies available to us, I will still use revenue as an extremely important feature going forward from the remaining 7000 movies.

The budget feature has some unclean values that makes Pandas assign it as a generic object. I proceed to convert this into a numeric variable and replace all the non-numeric values with NaN. Finally, as with budget, I will convert all the values of 0 with NaN to indicate the absence of information regarding budget.

Some of the columns that does not have an numeric value dropped from the dataset.

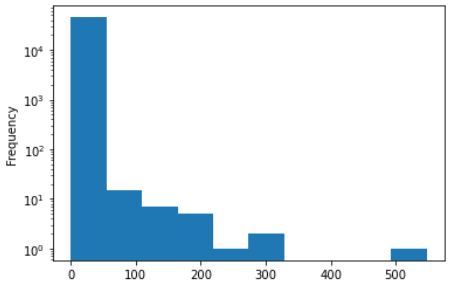
Some absence of dataset will be filled by median or mean :

```
sns.distplot(df['popularity'].fillna(df['popularity'].median()))
plt.show()
```



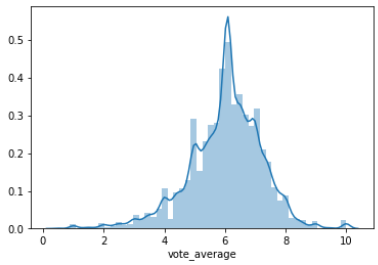

```
df['popularity'].plot(logy=True, kind='hist')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x78a815533dd0>
```



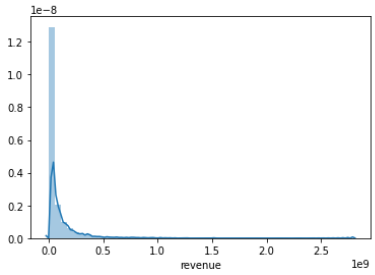
```
sns.distplot(df['vote_average'].fillna(df['vote_average'].median()))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x78a8152c3e10>
```




```
sns.distplot(df[df['revenue'].notnull()]['revenue'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x78a8141a1250>
```



I will perform the following feature engineering tasks:

1. Round up popularity values to 2 decimal values and I will be converting those into integer datatypes.

2. I will be creating dummy variables for column: genres

```
#count of each genre in the dataset
df['genre_ed'].value_counts()
```

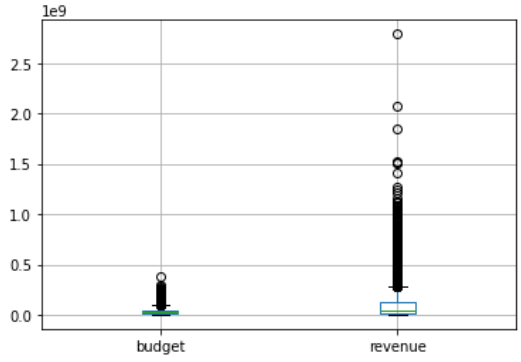
```
Drama              1313
Comedy             1067
Action              960
Adventure           416
Horror              325
Crime               267
Thriller            199
Animation           146
Fantasy             140
Romance             121
Science Fiction     104
Mystery              67
Family               55
Documentary          47
War                  38
Music                34
Western              31
History              29
Foreign               4
TV Movie              1
Name: genre_ed, dtype: int64
```

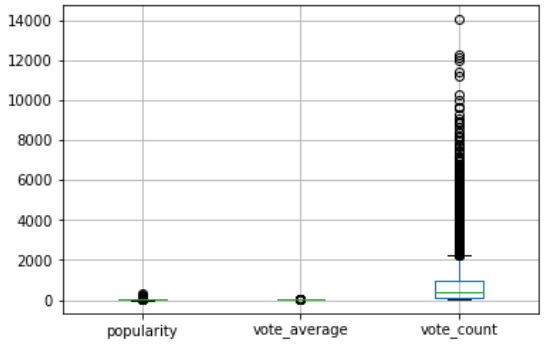
The dataset has most rows for Drama and Comedy. I will concentrate only gneres which has more than 100 rows.

After applying dummies :

```
Data columns (total 17 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   budget                  4255 non-null   float64
 1   popularity              4255 non-null   float64
 2   revenue                 4255 non-null   float64
 3   vote_average            4255 non-null   float64
 4   vote_count              4255 non-null   float64
 5   return                  4255 non-null   float64
 6   genre_ed_Action         4255 non-null   uint8
 7   genre_ed_Adventure      4255 non-null   uint8
 8   genre_ed_Animation      4255 non-null   uint8
 9   genre_ed_Comedy         4255 non-null   uint8
 10  genre_ed_Crime          4255 non-null   uint8
 11  genre_ed_Drama          4255 non-null   uint8
 12  genre_ed_Fantasy        4255 non-null   uint8
 13  genre_ed_Horror         4255 non-null   uint8
 14  genre_ed_Romance        4255 non-null   uint8
 15  genre_ed_Science Fiction  4255 non-null   uint8
 16  genre_ed_Thriller       4255 non-null   uint8
dtypes: float64(6), uint8(11)
memory usage: 278.4 KB
```

Some boxplot for analyzing the outliers :





PCA

Principal Component Analysis (PCA) is a dimensionality reduction technique commonly used in machine learning, statistics, and data analysis. The primary goal of PCA is to transform a high-dimensional dataset into a lower-dimensional representation while preserving as much of the original variance as possible. This reduction in dimensionality helps simplify the dataset, making it easier to analyze, visualize, and model.

Here is a brief description of the PCA algorithm:

1. Standardization of Data:

   - PCA is sensitive to the scale of the data. Therefore, it's important to standardize the features (subtract mean and divide by standard deviation) to give them a comparable scale.

2. Calculate the Covariance Matrix:

   - The covariance matrix is computed based on the standardized features. It represents the relationships between different features, indicating how they vary together.

3. Compute Eigenvectors and Eigenvalues:

   - The eigenvectors and eigenvalues of the covariance matrix are calculated. Eigenvectors represent the directions of maximum variance in the data, and eigenvalues indicate the magnitude of variance in those directions.

4. Sort Eigenvectors by Eigenvalues:

   - The eigenvectors are sorted based on their corresponding eigenvalues in descending order. This step is crucial as it determines the importance of each principal component. The higher the eigenvalue, the more variance is captured by the corresponding eigenvector.

5. Select Principal Components:

   - The top-k eigenvectors are chosen to form the principal components, where k is the desired dimensionality of the reduced dataset. These principal components represent the new coordinate system.

6. Create the Projection Matrix:

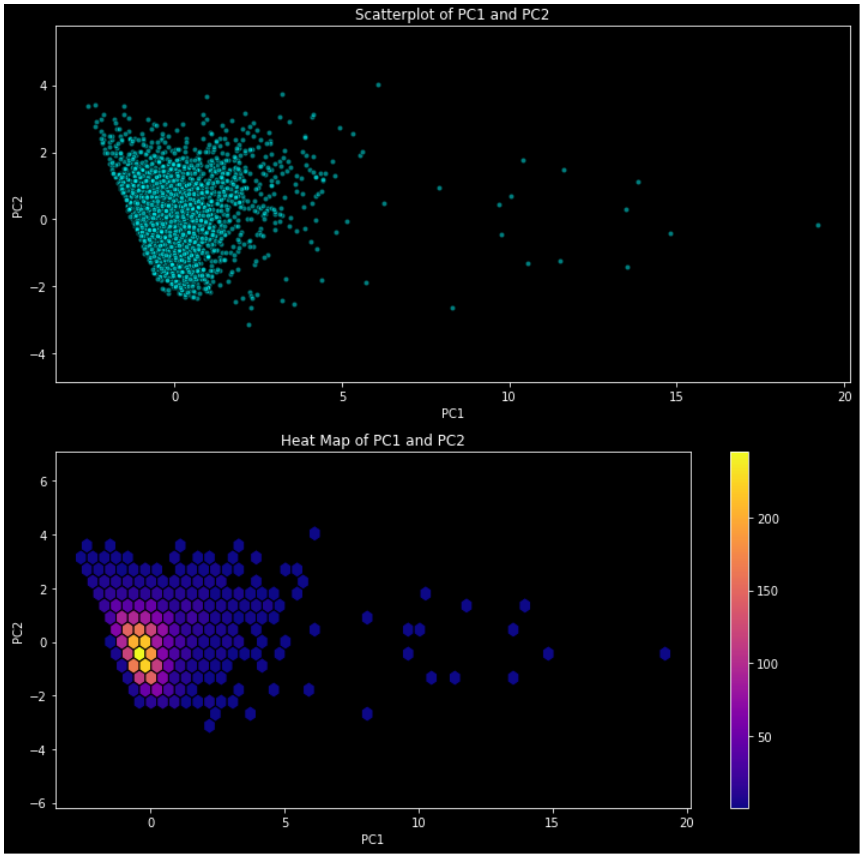
- The selected eigenvectors are arranged into a projection matrix. Multiplying the original data by this matrix yields the transformed, lower-dimensional representation.

7. Transform Data:

- The original dataset is transformed into the reduced-dimensional space using the projection matrix. Each data point is represented by a combination of the selected principal components.

PCA is widely used for various purposes, including noise reduction, feature extraction, and visualization of high-dimensional data. It is particularly useful in scenarios where a large number of features contribute to the dataset's complexity, and a more compact representation is desirable.

After using this algorithm our plot and heatmap:



Models

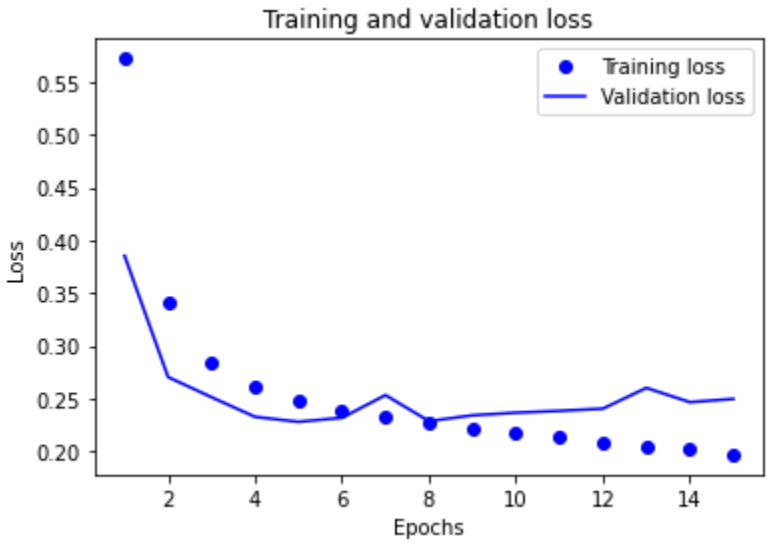
Neural Network

I have implemented three different neural networks to observe the effects of dropout and other factors. The results and details of these models are as follows:

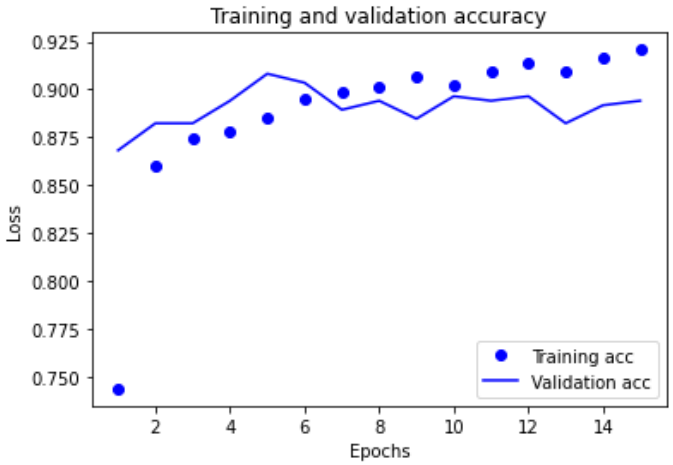
MLP 1 :

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 50)                850
_________________________________________________________________
dense_1 (Dense)              (None, 25)                1275
_________________________________________________________________
dense_2 (Dense)              (None, 15)                390
_________________________________________________________________
dense_3 (Dense)              (None, 10)                160
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 11
=================================================================
Total params: 2,686
Trainable params: 2,686
Non-trainable params: 0
```

Training and validation plot :



the training and validation loss plot is a valuable tool for understanding the training dynamics of a neural network, ensuring it learns from the data and generalizes well to new examples.



At the last classification report of this model (without any dropout) :

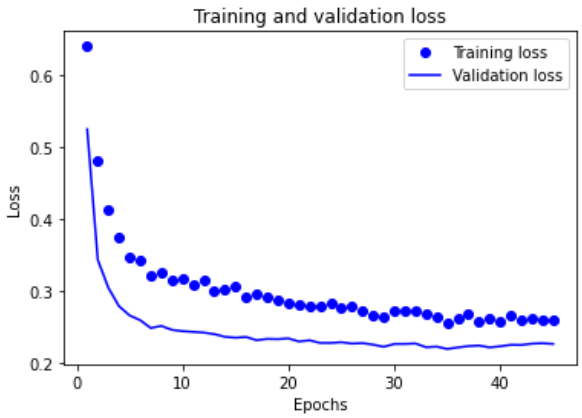
	precision	recall	f1-score	support
0	0.92	0.89	0.91	231
1	0.87	0.91	0.89	194
accuracy			0.90	425
macro avg	0.90	0.90	0.90	425
weighted avg	0.90	0.90	0.90	425

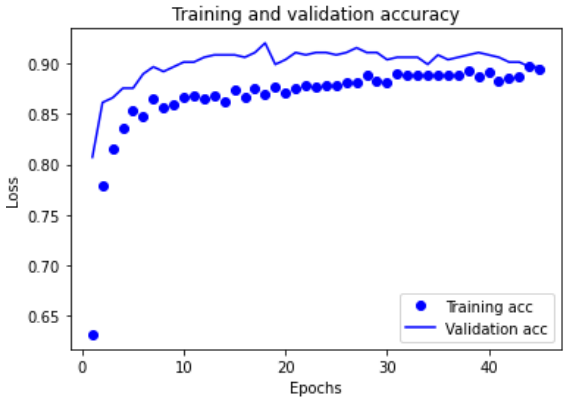
MLP 2 :

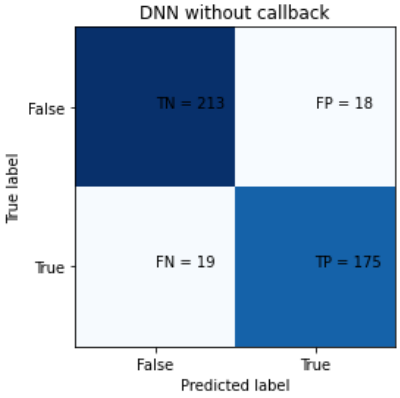
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_5 (Dense)              (None, 50)                850
_________________________________________________________________
dropout (Dropout)            (None, 50)                0
_________________________________________________________________
dense_6 (Dense)              (None, 25)                1275
_________________________________________________________________
dropout_1 (Dropout)          (None, 25)                0
_________________________________________________________________
dense_7 (Dense)              (None, 10)                260
_________________________________________________________________
dropout_2 (Dropout)          (None, 10)                0
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 11
=================================================================
Total params: 2,396
Trainable params: 2,396
Non-trainable params: 0
```

In this model, we have 3 dropout layer with with the probability of 0.2, 0.3 and 0.4 in order. In neural networks, dropout is a regularization technique that randomly deactivates a fraction of neurons during training. This introduces noise, prevents overfitting, and promotes a more generalized model. Dropout has an ensemble effect, improving the model's ability to generalize to new data. It is implemented after activation functions, and during testing,

dropout is turned off. Overall, dropout enhances robustness, reduces sensitivity to specific neurons, and contributes to better model performance.







A confusion matrix is an evaluation tool in machine learning used for classification problems. It represents the number of instances of each class and how the model assigns them. The main components of a confusion matrix include:

True Positive (TP): The number of instances where the model correctly predicts the positive class.

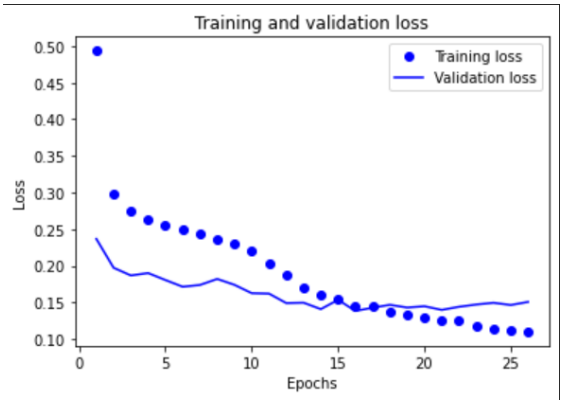
True Negative (TN): The number of instances where the model correctly predicts the negative class.

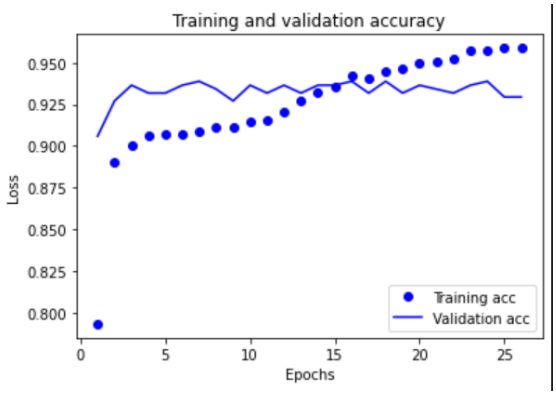
False Positive (FP): The number of instances where the model incorrectly predicts the positive class (Type I error).

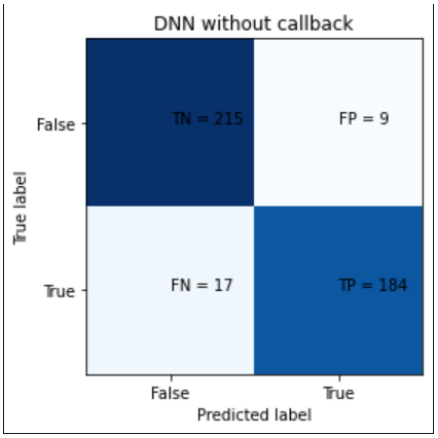
False Negative (FN): The number of instances where the model incorrectly predicts the negative class (Type II error).

MLP 3 (without dropout):

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_9 (Dense)              (None, 20)                340
_________________________________________________________________
dense_10 (Dense)             (None, 50)                1050
_________________________________________________________________
dense_11 (Dense)             (None, 50)                2550
_________________________________________________________________
dense_12 (Dense)             (None, 20)                1020
_________________________________________________________________
dense_13 (Dense)             (None, 1)                 21
=================================================================
Total params: 4,981
Trainable params: 4,981
Non-trainable params: 0
_________________________________________________________________
```







KNN : K nearest neighbour

K-Nearest Neighbors (KNN) is a simple algorithm for classification and regression. It predicts a new data point's label or value by considering the majority class or average of its K-nearest neighbors in the feature space. The choice of K is a critical factor, and scaling features is recommended. KNN is straightforward to implement but may be sensitive to the dataset's characteristics.

I iterate over the between 3 to 25 for K and the best accuracy that it reached was in the k = 23.

SVM

Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. In the context of classification, SVM aims to find a hyperplane that best separates the data into different classes while maximizing the margin between the classes. The margin is the distance between the hyperplane and the nearest data points from each class. SVM is effective in high-dimensional spaces and is particularly well-suited for cases where the classes are not linearly separable.

The key concepts in SVM include:

1. Hyperplane: In a two-dimensional space, a hyperplane is a line that separates the data into two classes. In higher-dimensional spaces, it becomes a multidimensional plane.

2. Support Vectors: These are the data points that lie closest to the decision boundary (hyperplane). They play a crucial role in determining the optimal hyperplane.

3. Margin: The margin is the distance between the hyperplane and the nearest data point from each class. SVM aims to maximize this margin.

4. Kernel Trick: SVM can efficiently handle non-linear decision boundaries through the use of a kernel function. This function transforms the input features into a higher-dimensional space, making it possible to find a hyperplane in that space.

5. C Parameter: This parameter in SVM controls the trade-off between achieving a smooth decision boundary and classifying the training points correctly. It helps prevent overfitting.

SVM is versatile and can be applied to various types of data. It is widely used in image classification, text categorization, and bioinformatics, among other fields.

To achieve the best performance in our model, we utilized the **RandomizedSearchCV** library to fine-tune the hyperparameters. This approach allowed us to systematically search through different hyperparameter combinations and identify the optimal configuration for our SVM model. The goal was to obtain the best possible result and report it as the optimal outcome achieved through the SVM model.

```
[CV]  C=5.694898104415122, gamma=0.042619372215805924, kernel=linear, total=   0.5s
[CV] C=5.694898104415122, gamma=0.042619372215805924, kernel=linear ..
[CV]  C=5.694898104415122, gamma=0.042619372215805924, kernel=linear, total=   0.5s
[CV] C=2.444729736792577, gamma=0.06486844010524996, kernel=poly .....
[CV]  C=2.444729736792577, gamma=0.06486844010524996, kernel=poly, total=   0.1s
[CV] C=2.444729736792577, gamma=0.06486844010524996, kernel=poly .....
[CV]  C=2.444729736792577, gamma=0.06486844010524996, kernel=poly, total=   0.1s
[CV] C=2.444729736792577, gamma=0.06486844010524996, kernel=poly .....
[CV]  C=2.444729736792577, gamma=0.06486844010524996, kernel=poly, total=   0.1s
[CV] C=1.9687976633732798, gamma=0.015975906128302957, kernel=rbf ....
[CV]  C=1.9687976633732798, gamma=0.015975906128302957, kernel=rbf, total=   0.1s
[CV] C=1.9687976633732798, gamma=0.015975906128302957, kernel=rbf ....
[CV]  C=1.9687976633732798, gamma=0.015975906128302957, kernel=rbf, total=   0.1s
[CV] C=1.9687976633732798, gamma=0.015975906128302957, kernel=rbf ....
[CV]  C=1.9687976633732798, gamma=0.015975906128302957, kernel=rbf, total=   0.1s
[CV] C=8.1195523353288, gamma=0.03181926147052891, kernel=poly .......
[CV]  C=8.1195523353288, gamma=0.03181926147052891, kernel=poly, total=   0.1s
[CV] C=8.1195523353288, gamma=0.03181926147052891, kernel=poly .......
[CV]  C=8.1195523353288, gamma=0.03181926147052891, kernel=poly, total=   0.1s
[CV] C=8.1195523353288, gamma=0.03181926147052891, kernel=poly .......
[CV]  C=8.1195523353288, gamma=0.03181926147052891, kernel=poly, total=   0.1s
[CV] C=8.044981573568046, gamma=0.004800089988044231, kernel=poly ....
[CV]  C=8.044981573568046, gamma=0.004800089988044231, kernel=poly, total=   0.2s
[CV] C=8.044981573568046, gamma=0.004800089988044231, kernel=poly ....
[CV]  C=8.044981573568046, gamma=0.004800089988044231, kernel=poly, total=   0.3s
[CV] C=8.044981573568046, gamma=0.004800089988044231, kernel=poly ....
[CV]  C=8.044981573568046, gamma=0.004800089988044231, kernel=poly, total=   0.2s
[CV] C=7.883166608513233, gamma=0.007207859353882385, kernel=rbf .....
[CV]  C=7.883166608513233, gamma=0.007207859353882385, kernel=rbf, total=   0.1s
[CV] C=7.883166608513233, gamma=0.007207859353882385, kernel=rbf .....
[CV]  C=7.883166608513233, gamma=0.007207859353882385, kernel=rbf, total=   0.1s
[CV] C=7.883166608513233, gamma=0.007207859353882385, kernel=rbf .....
[CV]  C=7.883166608513233, gamma=0.007207859353882385, kernel=rbf, total=   0.1s
[CV] C=3.9221584848645143, gamma=0.02302270715631461, kernel=rbf .....
[CV]  C=3.9221584848645143, gamma=0.02302270715631461, kernel=rbf, total=   0.1s
[CV] C=3.9221584848645143, gamma=0.02302270715631461, kernel=rbf .....
[CV]  C=3.9221584848645143, gamma=0.02302270715631461, kernel=rbf, total=   0.1s
```

At last, the best hyperparameters is :

```
Best estimator:  SVC(C=3.9221584848645143, gamma=0.023022707156314 61)
test accuracy:  0.9317647058823529
MSE :  0.06823529411764706
```

You can see the accuracy and its error in the above image.

Random Forest

Random Forest is an ensemble learning algorithm that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Here are some key details about the Random Forest model:

1. Ensemble of Decision Trees: Random Forest builds multiple decision trees during training and merges them together to get a more accurate and stable prediction.

2. Random Feature Selection: At each split in a decision tree, a random subset of features is considered for splitting. This randomness helps to decorrelate the trees and make the model more robust.

3. Bagging Technique: The algorithm uses a technique called bagging (Bootstrap Aggregating), where each tree is trained on a random subset of the training data, with replacement.

4. Robust to Overfitting: Random Forest is less prone to overfitting compared to individual decision trees. The ensemble of diverse trees tends to generalize well to unseen data.

5. Feature Importance: Random Forest provides a measure of the importance of each feature in making accurate predictions, allowing for feature selection.

6. Parallel Training: The individual trees in a Random Forest can be trained in parallel, making it a scalable algorithm for large datasets.

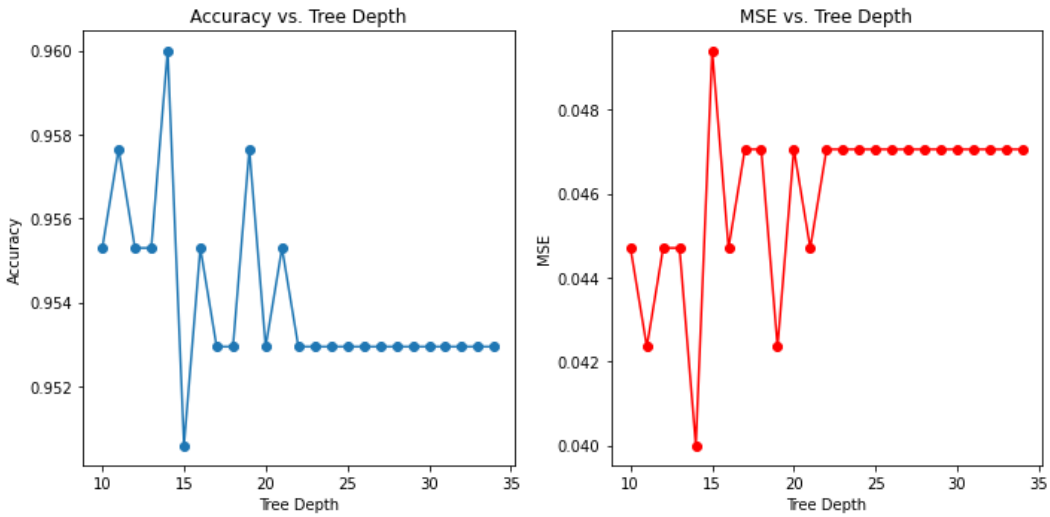
7. Versatility: Random Forest can be used for both classification and regression tasks.

8. Hyperparameters: Important hyperparameters include the number of trees in the forest, the depth of the trees, and the number of features considered for each split.

9. Out-of-Bag (OOB) Error: Random Forest uses out-of-bag samples (data not used in a tree's training) to estimate the model's performance without the need for a separate validation set.

Overall, Random Forest is a powerful and versatile machine learning algorithm suitable for a variety of tasks, offering robustness and high predictive accuracy.

In this model, the results are examined based on the depth of decision trees. It can be observed that after a depth of 22, the accuracy and MSE results remain constant and do not change. However, the best result is achieved at a depth of 14 with an accuracy of 96%.



Radial Basis Neural Network

Radial Basis Function (RBF) Neural Network is a type of artificial neural network that uses radial basis functions as activation functions. Unlike traditional neural networks that typically use sigmoid or rectified linear unit (ReLU) activation functions, RBF networks use radial basis functions to model complex relationships in the data.

The basic architecture of an RBF neural network includes three layers: an input layer, a hidden layer with radial basis functions, and an output layer. Here's a brief explanation of each layer:

1. **Input Layer:** This layer consists of nodes representing the input features. Each node corresponds to a feature in the input data.
2. **Hidden Layer:** The hidden layer of an RBF network is unique because it employs radial basis functions as activation functions. These functions are centered at specific points in the input space and measure the similarity between the input data and these centers. Common radial basis functions include Gaussian functions.
3. **Output Layer:** The output layer produces the final result of the network. The nodes in this layer combine the information learned by the radial basis functions to generate the output.

Training an RBF neural network involves determining the optimal centers and widths of the radial basis functions. This is often done using techniques such as k-means clustering. Once trained, the network can make predictions on new, unseen data.

RBF neural networks are particularly useful for problems with complex, non-linear relationships. They have been applied in various fields, including pattern recognition, time-series prediction, and function approximation.

In this implementation, firstly I implement RBF layer class to use it like Dense in keras.sequential and use it for predicting the classification.

RBF layer :

```
from keras.layers import Layer
from keras import backend as K

class RBFLayer(Layer):
    def __init__(self, units, gamma, **kwargs):
        super(RBFLayer, self).__init__(**kwargs)
        self.units = units
        self.gamma = K.cast_to_floatx(gamma)

    def build(self, input_shape):
        self.mu = self.add_weight(name='mu',
                                  shape=(int(input_shape[1]), self.units),
                                  initializer='uniform',
                                  trainable=True)
        super(RBFLayer, self).build(input_shape)

    def call(self, inputs):
        diff = K.expand_dims(inputs) - self.mu
        l2 = K.sum(K.pow(diff,2), axis=1)
        res = K.exp(-1 * self.gamma * l2)
        return res

    def compute_output_shape(self, input_shape):
        return (input_shape[0], self.units)
```

The second MLP has the best accuracy, so we implement RBF using RBF layer with the Radial Basis activation functions.

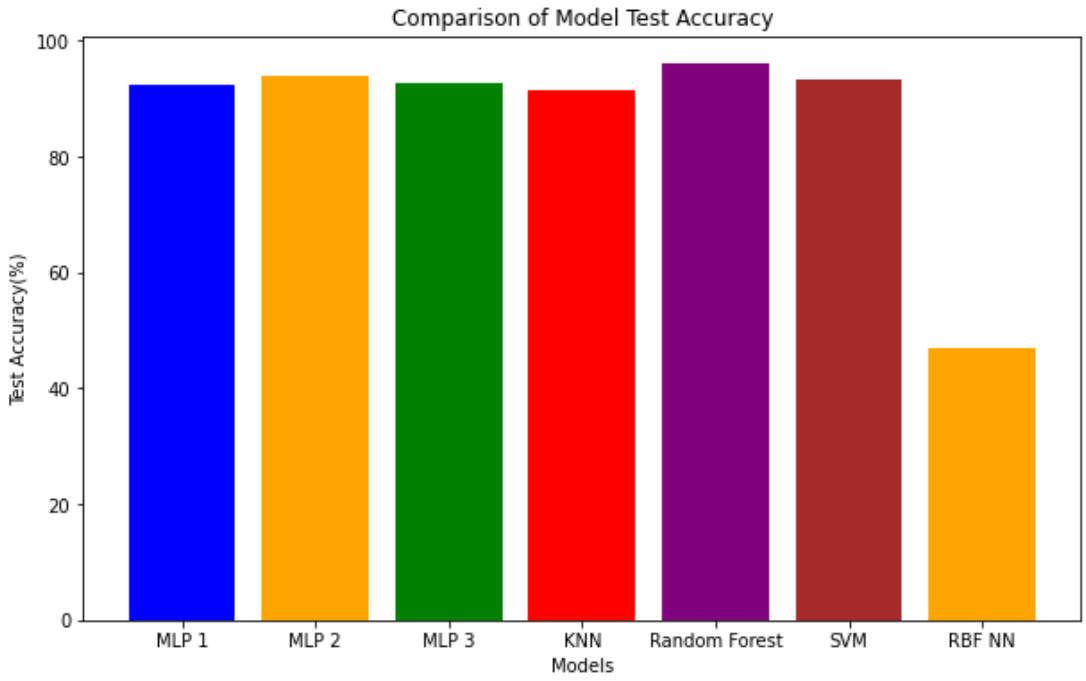
```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rbf_layer_10 (RBFLayer)      (None, 50)                800
_________________________________________________________________
dropout_14 (Dropout)         (None, 50)                0
_________________________________________________________________
rbf_layer_11 (RBFLayer)      (None, 25)                1250
_________________________________________________________________
dropout_15 (Dropout)         (None, 25)                0
_________________________________________________________________
rbf_layer_12 (RBFLayer)      (None, 10)                250
_________________________________________________________________
dropout_16 (Dropout)         (None, 10)                0
_________________________________________________________________
rbf_layer_13 (RBFLayer)      (None, 1)                 10
=================================================================
Total params: 2,310
Trainable params: 2,310
Non-trainable params: 0
_________________________________________________________________
```

After the testing predictions of this model, we have found that this model can not have a good prediction on this data because it doesn't have circular distribution.

At the end, We have gathered the accuracies and MSE of 7 different models and show it as a data frame :

	Model	Test Accuracy(%)	Mean squared Error
0	MLP 1	92.470588	0.075294
1	MLP 2	93.882353	0.061176
2	MLP 3	92.705882	0.072941
3	KNN	91.294118	0.087059
4	Random Forest	96.000000	0.040000
5	SVM	93.176471	0.068235
6	RBF NN	46.823529	0.531765

And a plot for it :



These plots depict that the Random Forest (RF) model achieved the best performance among all the employed modeling techniques with 96% accuracy. Among the neural networks used, the model incorporating dropout layers outperformed others, demonstrating superior performance with an accuracy rate of 93.88%, securing the second position. This highlights the significance of utilizing dropout layers in neural network architectures and the effectiveness of the specific neural network configuration.