Titanic Survival Analysis Using Machine Learning

Introduction

Problem statement:

The sinking of the Titanic stands as one of the most tragic disasters in history.
Regrettably, the insufficient number of lifeboats led to the loss of many lives. Survival on that fateful night seemed to involve an element of chance, yet certain factors may have played a role in getting rescued.
The aim is to construct a predictive model to find if certain factors influenced the chances of survival of passengers. This involves utilizing passenger data such as ages, sex, travel classes, and other relevant factors.

Dataset:

The dataset consists of details of passengers on titanic. It has 12 columns and 891 rows on the training dataset.
The columns indicate the following details about the passenger:

PassengerID: identification number
Survived: 0-passenger survived and 1-passenger died
Pclass: The class in which the passenger was travelling.
Name
Sex: Male or female
Age
sibsp : The dataset defines family relations in this way.
Sibling = brother, sister, stepbrother, stepsister
Spouse = husband, wife (mistresses and fiancés were ignored)
parch: The dataset defines family relations in this way...
Parent = mother, father
Child = daughter, son, stepdaughter, stepson
Some children travelled only with a nanny, therefore parch=0 for them.
Ticket
Fare
Cabin: Cabin number
Embarked: Port of Embarkation C = Cherbourg, Q = Queenstown, S = Southampton

Objective:

The objective of this project is to train 3 models from the following 3 categories: Fix-shape universal approximators (kernel methods), Neural network based universal approximators, Tree-based approaches, upon preprocessing the training dataset and deploying the highest performing model to make predictions regarding which passengers survived on the testing dataset.

Methodology

Preprocessing

In the training and testing data, We first print what all columns have null values: Cabin, Embarked, Fare, Age
Columns such as PassengerID, Name, Cabin, Ticket are mostly unique values and are not expected to influence the Survived parameter, hence these columns are dropped and will not be used to train the models to reduce complexity and to include only relevant data.
The null values are repraced appropriately

Categorical Variables like Sex and Embarked are one-hot encoded.
All numerical variables are standard normalized.

Model definition and Validation Testing

The data from train.csv file is preprocessed and split into training and testing sets
The models are defined and validation testing is done using Kfolds (k=5) and their mean accuracies are computed for validation testing.

Model descriptions and their hyperparameters:

SVM model:
From Fix-shape universal approximators, I have chosen SVM for the following reasons:
Effective in High-Dimensional Spaces: work well in high-dimensional spaces, making them suitable for datasets with 6 training variables. This is particularly beneficial when dealing with complex relationships among multiple features.

Hyper parameters:
Kernel method: rbf.
Reason: I tried all 4 kernels (polynomial, linear, sigmoid, and rbf) and achieved highest performance with rbf.

- RBF kernels are known for their ability to express complex relationships, enabling SVMs to approximate any continuous function. This makes them powerful universal approximators, suitable for a wide range of data distributions.

C(regularization parameter)=0.5
C is the regularization parameter that controls the trade-off between achieving a low training error and a low testing error. A smaller C encourages a larger-margin decision boundary but may

Gamma= auto
Gamma defines how far the influence of a single training example reaches.
I tried different small and large values of gamma, got the best result when it was set to auto.

Neural network based universal approximator model:
I have chosen a fully connected sequential Neural network.

Advantages:

- Neural networks, particularly with multiple layers, are capable of capturing complex, non-linear relationships in the data. If the relationships between your input variables and the binary target variable are intricate and cannot be effectively modeled by linear methods, a neural network can provide the flexibility needed.
- neural network with multiple layers allows the model to learn hierarchical representations of features. Each layer can capture different levels of abstraction, enabling the network to automatically extract relevant features from the input data.

Hyper parameters:
Architecture hyperparameters:
These define the structure of the neural network.

Layers=5: Specifies the total number of layers in the neural network. For a fully connected network, this includes input, hidden, and output layers.

Number of neurons in each layer: The choice of neurons in the hidden layers affects the network's capacity to learn complex patterns. The 5 layers have 2^8,2^7,2^6,2^5,2^0 neurons.

Activation function: ReLU in 1st four layers, Sigmoid in last layer (explain)
The choice of ReLU (Rectified Linear Unit) activation functions in the first four layers and a sigmoid activation function in the last layer is an optimum configuration for neural networks, especially in binary classification tasks.

Sigmoid Activation:
It squashes the input values to the range of [0, 1], making it suitable for binary classification problems. The sigmoid function is commonly used in the output layer of binary classification models because it can interpret the network's output as a probability.
Advantages:

- Outputs values between 0 and 1, representing probabilities.
- Smooth gradient, aiding in stable training.

Training Hyperparameters:
Learning rate: I have used decaying learning rate to avoid overfitting and make sure my model converges well.
Initial learning rate:0.01, decay steps=1000, decay rate=0.01
A higher initial learning rate (0.01 in this case) allows the model to take larger steps during optimization, potentially speeding up the convergence process. However, as training progresses, it is gradually reduced to converge more accurately towards the minimum of the loss function. A decaying learning rate adds a level of stability to the training process.

I have experimented with different learning rates, decay steps, and decay rates. The aforementioned values gave the best results.

Optimizer = Adam
The optimizer is the algorithm used to update the model's weights during training. I have manually experimented with the following optimizers: Adam, RMSprop, and SGD (Stochastic Gradient Descent). Adam gave the best results.

- Adam adjusts the learning rates for each parameter individually based on the historical gradient information. This adaptive learning rate helps accelerate convergence by allowing the model to take larger steps for less frequently updated parameters and smaller steps for frequently updated parameters.

Loss function: binary cross entropy
The loss function measures the difference between the model's predictions and the actual target values.
Reasons for choosing binary cross entropy:

- It is well-suited for models using the sigmoid activation function because it leverages the properties of the sigmoid function.
- It is robust when dealing with imbalanced datasets, where the number of examples in each class may differ significantly. It penalizes misclassifications in both classes proportionally, helping the model generalize better to imbalanced scenarios.

Epochs=1500
An epoch is a single pass through the entire training dataset. The number of epochs defines how many times the model will see the entire dataset during training. Epoch=1500 gave the best results

Batch size=64
Batch size determines the number of training examples used in each iteration of training. This value gave the best results

Random Forest

From tree-based approaches, I have chosen random forest
Random Forest is an ensemble learning method that builds multiple decision trees during training and merges their predictions during inference. This ensemble approach often results in improved predictive performance compared to individual decision trees.
Advantages:

- It is known for their high predictive accuracy. They can handle both numerical and categorical features, making them versatile for various types of datasets. Even with a relatively small number of training variables (6-7 in your case), Random Forests can capture complex relationships in the data.
- Less prone to overfitting compared to individual decision trees, especially when the number of trees in the forest is reasonable. The ensemble nature of Random Forests helps mitigate overfitting by aggregating predictions from multiple trees.

Hyperparameters:
Max depth=4
The maximum depth of each decision tree in the forest. I tested for different values of depth 4 gave the best performance.

Number of estimators=200
The number of decision trees in the forest. This value gave best results.

Ccp_alpha=0.01
The complexity parameter (α) used for Minimal Cost-Complexity Pruning. It controls the amount of regularization applied to the tree. Tried various values for alpha. 0.01 gave best results
random_state: 42
A seed for reproducibility. When set to a specific value, the results will be reproducible across different runs.

Training

After the best hyperparameters are selected after validation testing, the three models are run on the entire train split set and their accuracies are evaluated and compared.

Testing and performance comparison

The performance of all three models is compared on the test split set. The performance metrics include accuracy, precision, recall, and f1 score, and visualizations include a bar graph comparing accuracy, precision, recall and f1 scores for all 3 models, ROC curve,

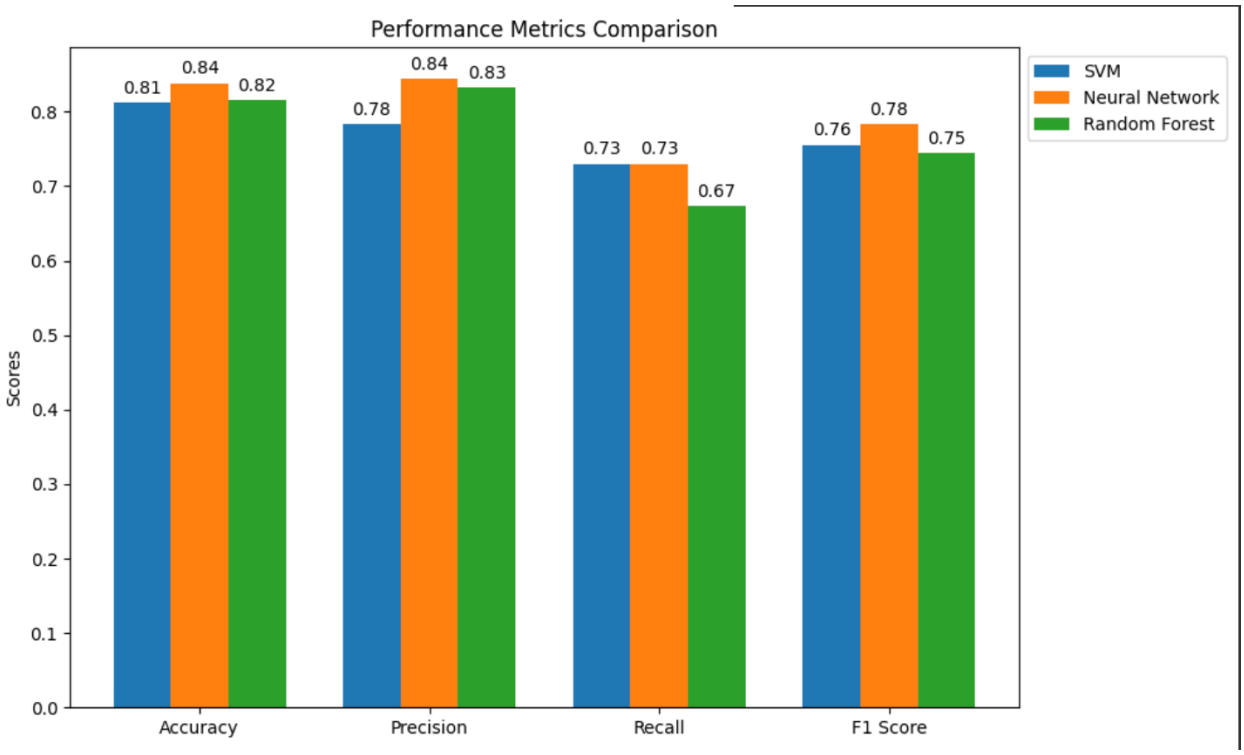
Accuracy: The proportion of correctly classified instances among the total instances.
Ranking models on the basis of accuracy: NN>RF>SVM

Precision: The ratio of true positive predictions to the total predicted positives.
Ranking models on the basis of precision: NN>RF>SVM

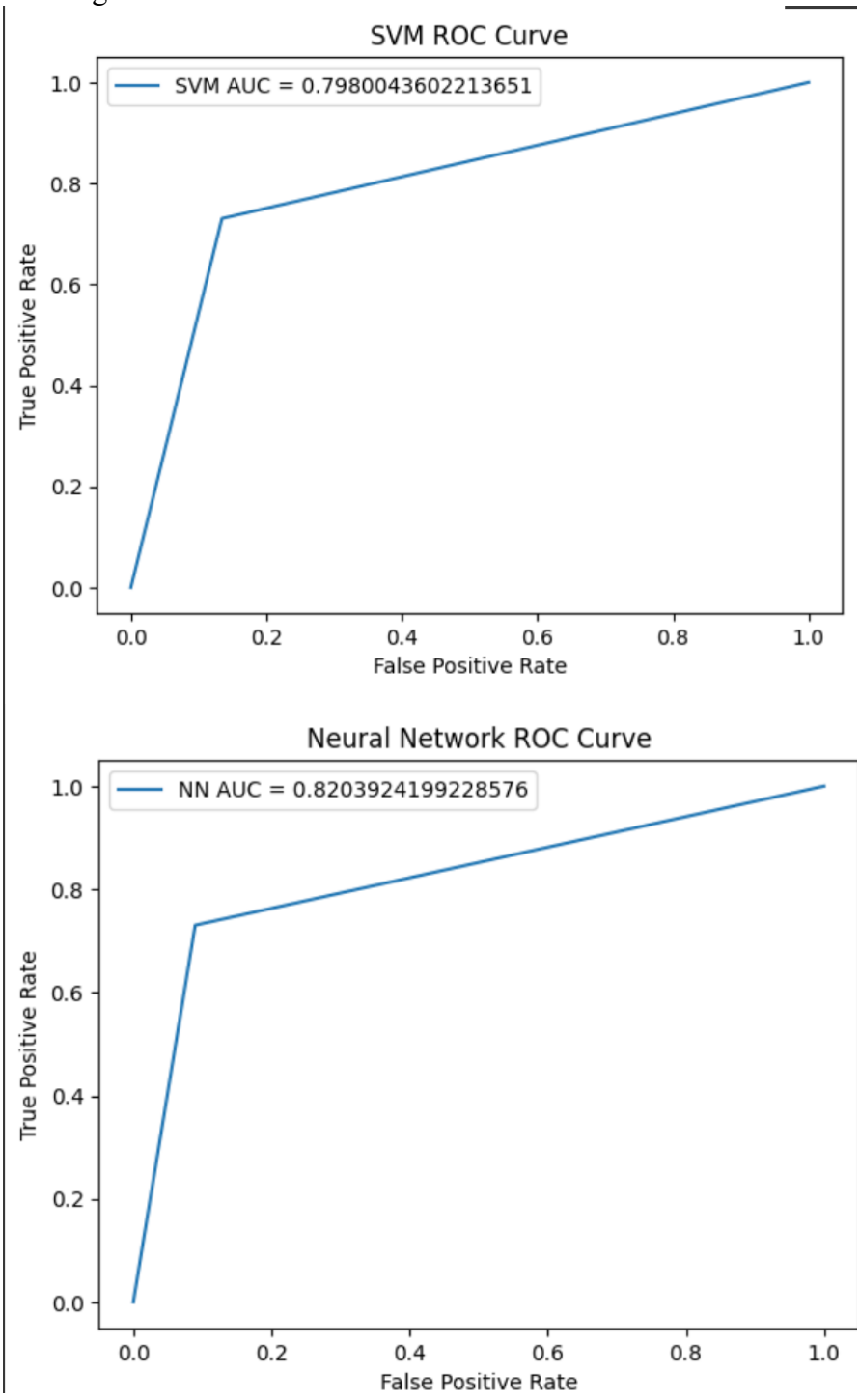
*Recall (Sensitivity or True Positive Rate):*The ratio of true positive predictions to the total actual positives.

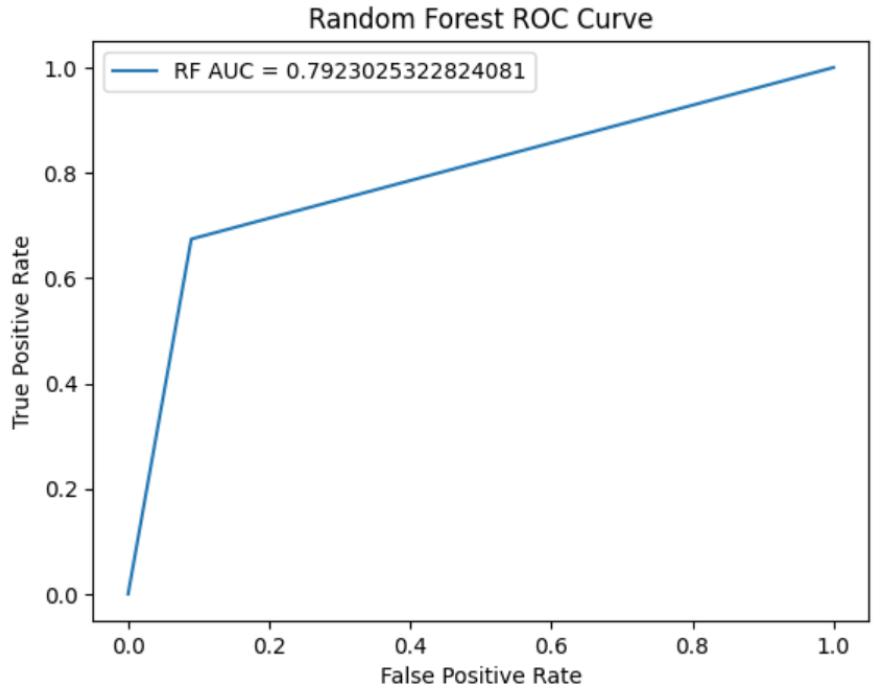
Ranking models on the basis of Recall score: NN=SVM>RF

F1 Score: The harmonic mean of precision and recall.F1 score provides a balanced measure between precision and recall.
Ranking models on the basis of F1 score: NN>SVM>RF



ROC Curve (Receiver Operating Characteristic Curve):
It is graphical representation of the trade-off between true positive rate (sensitivity) and false positive rate at various thresholds.
Ranking the models on the basis of area under ROC curve: NN >SVM>RF

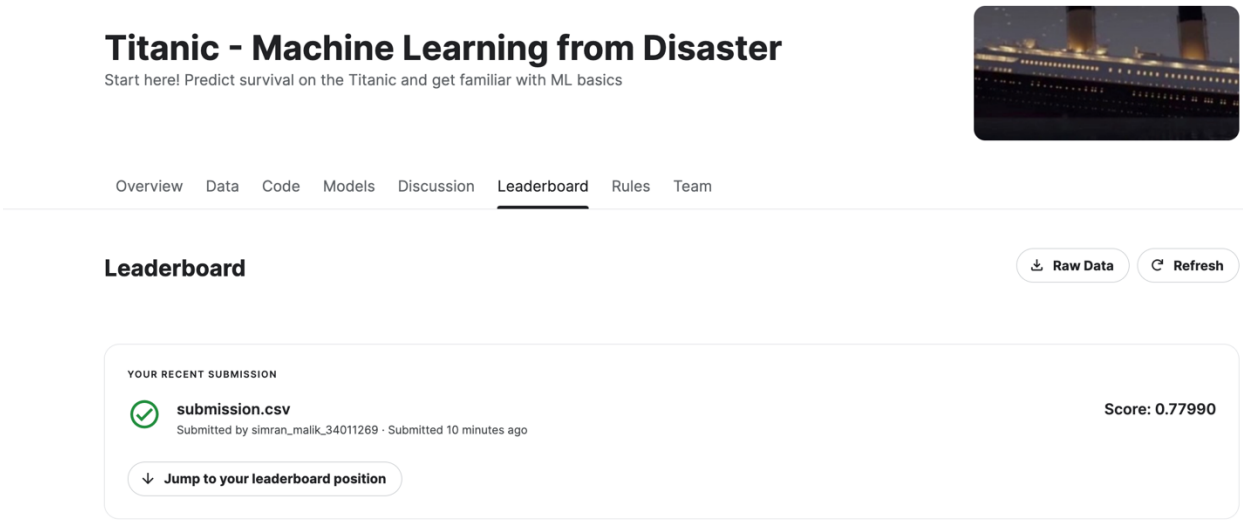




Making Prediction on the test.csv using best performing model

Neural Network is the best performing model. It is then used to make predictions of the test.csv file. The predictions are then mapped with their corresponding Passenger ids, and the output is saved in a csv file, which is uploaded to Kaggle.

The accuracy score on Kaggle is: 0.7799 ~ 0.78



Conclusion

Neural network is the best performing model across all performance metrics.
Neural Network Outperformance:
The neural network (NN) has consistently outperformed both Support Vector Machine (SVM) and Random Forest (RF) across multiple evaluation metrics, including accuracy, precision, recall, and F1 score.

Potential areas for improvement:

- The three models can be tuned better by experimenting further hyperparameter tuning.
- Additional feature engineering techniques can be applied to enhance the models' ability to capture relevant patterns.
- Increasing the size of the training dataset may help in enhancing the models performance