PROJECT FINAL REPORT

APPYLING MACHINE LEARNING ON TITANIC

Vishal Addala(vxa162530)

Sai Anuhya Devalla(sxd161030)

Dinesh Chalasani(dxc163530)

Srivallika Medandarao(sxm165730)

TABLE OF CONTENTS

INTRODUCTION:

"RMS Titanic", Known to the world as Grandeur Ship at that time before disaster and was the most spectacular ship at that time. The wreckage of the Gigantic Titanic Ship is one of the worst disaster that world has witnessed in Sea Voyage history. On April 15, 1912, The Titanic collided with an iceberg near Newfoundland, Canada and sunk killing 1502 out of 2224 passengers and crew.

The public inquiries and government analysis reports revealed that this was due to many regulatory and operational failures. Out of which, insufficient lifeboats was one of the most criticized, which led to more deaths. The Titanic only carried enough lifeboats for around 1178 people out of total 2224 passengers.

This kind of analysis has caught attention of Data Scientist's and Machine Learning Engineer's to discover new hidden insights and Patterns that can reveal about the chance of a person to survive with respective to Class, Age, Gender, etc... In this project we applied multiple many learning algorithms and built a model that can predict the chance of survival.

PROBLEM STATEMENT:

The goal of our project is to predict whether a given passenger survived or not given his other details. To achieve this, we implement several machine learning classifiers and find out which classifier gives us the best prediction. With great interest in machine learning prediction techniques, we would like to identify the patterns in the passenger survival data and develop a prediction model to predict the chance of a passenger in surviving the tragedy based on the information/ variables.

DATASET DETAILS:

Total three datasets are provided by Kaggle, which are Test.CSV, Train.CSV, geneder_submission.CSV. Total Test and Train dataset had 1309 instances with 11 variables. Out of all these variables, 'passengerID' is the unique identifier across all datasets and evaluations or submissions are based on predicting the chance of survival for these passengers. The prediction models will be trained from the 'training set' which will be split into training and test set.Our model is applied to the test set and the results are used to evaluate the model or in other words, how well our model is performing on unseen dataset.

- VARIABLES:

As mentioned earlier, the following are the variables that are considered for predicting the survival of the passenger in the tragedy.

Variable Names	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	1 = Upper, 2 = Middle, 3 = Lower
sex	Sex	

Age	Age in years	
sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Notes:
age: Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5

parch: if children travelled only with a nanny, then parch=0.

Evaluation Metrics: We would be using the following metrics to evaluate our models:

1)Precision

2)F-score

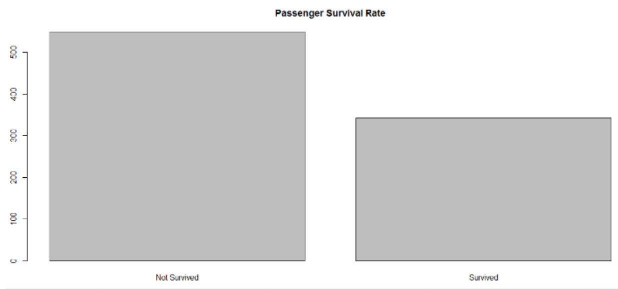
3)ROC Curve

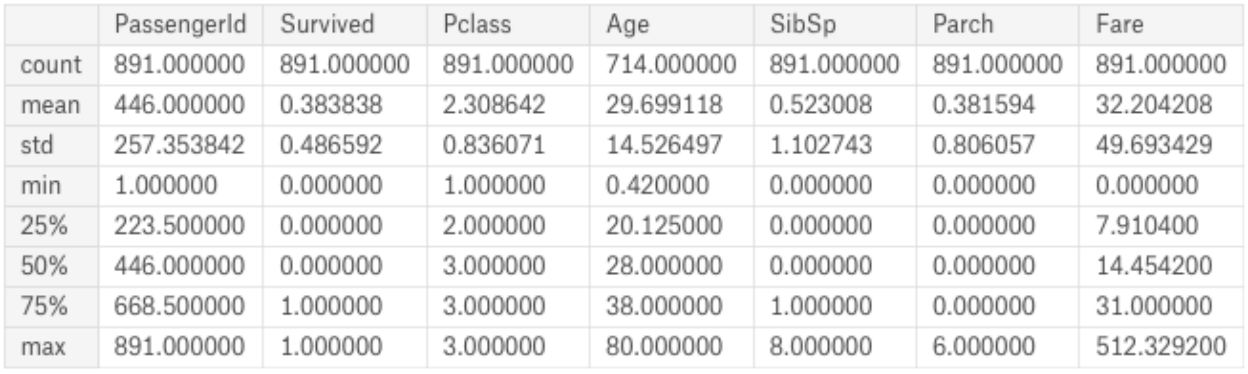
4)ROC AUC (area under the curve)

EXPLORATION DATA ANALYSIS:

The following are the observations made after reviewing the summary statistics (FIGURE 1)

- Total no of elements in training dataset is 891
- The mean age of the passengers was 29.66, youngest passenger was 0.42 years and the eldest was 80 years
- Average fare of a ticket was 32.20





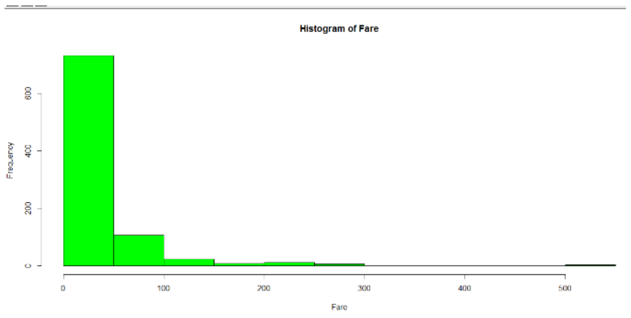
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

FIGURE 1

In the process of exploring the dataset, we performed univariate analysis on the individual variables.

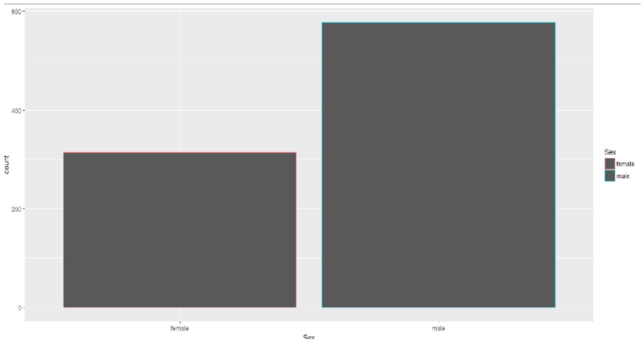
From the visualization, we notice that there are more people dies ('Not Survived') than survived.

FIGURE 2. *Count of Passengers Survived or Not*



The "Fare" variable data distribution illustrates a positive skew distribution with more people on board had low fare tickets.

FIGURE 3. *Fare Variable data distribution*



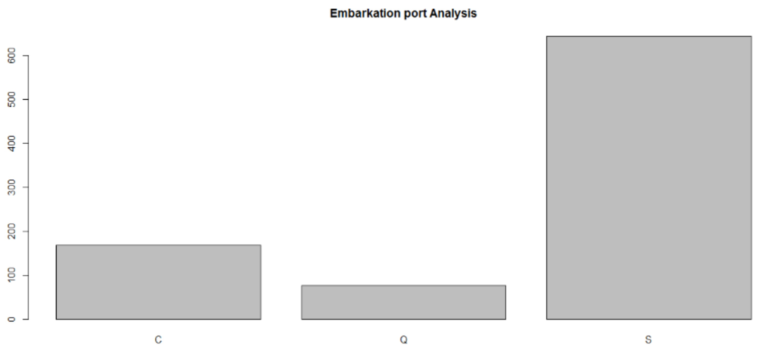
The Plot illustrates that Titanic had more Male head count than Female.

FIGURE 4

The plot illustrates that more people boarded on **Southampton ('S')**

FIGURE 5

From the exploatoty analysis, we can notice that there are more passengers on board whose ages are between 17 to 35.



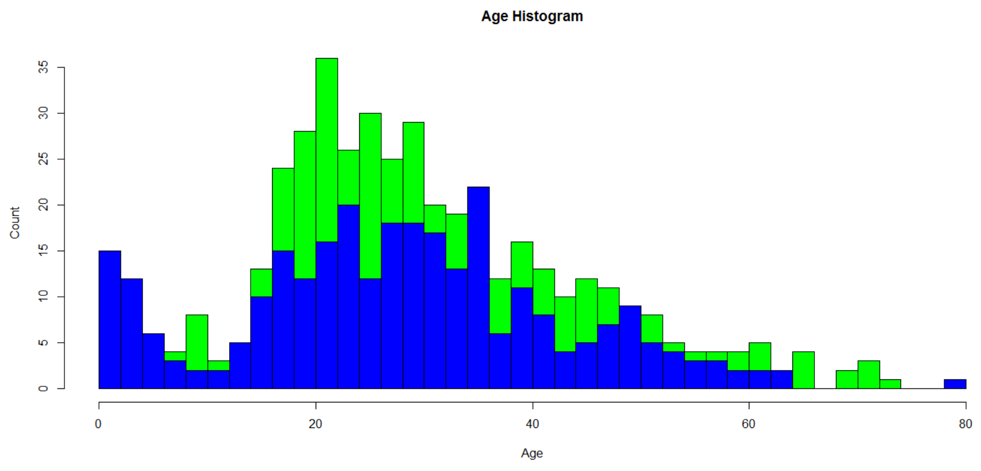


FIGURE 6

Now, Exploring Correlation of Surviving with respect to other variables. we analyzed the dependent variables with Independent variable “Survived” on training dataset. The following plot represents the each of the three ports of embarkment S, C, G with the percentage of survivals based on Sex.

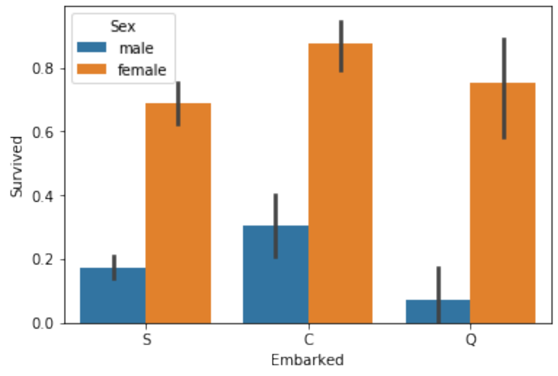


FIGURE 7

From the plot, it clearly illustrates that, the percentage of female passengers surviving the tragedy is higher than male passengers.

Similarly, analysis of dependent variable ‘Pclass’ with independent variable ‘Survived’ illustrated that the chance of survival reduced from High class (Pclass = 1) to low class (Pclass = 3). Second conclusion is that, the chance of male in Pclass 2 & 3 is very slim when compared with others.

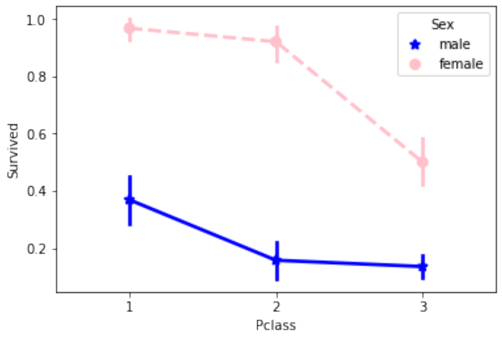


FIGURE 8

The graph illustrates the distribution of the age column across the survived column and we can conclude that babies had more chance of survival than other age groups.

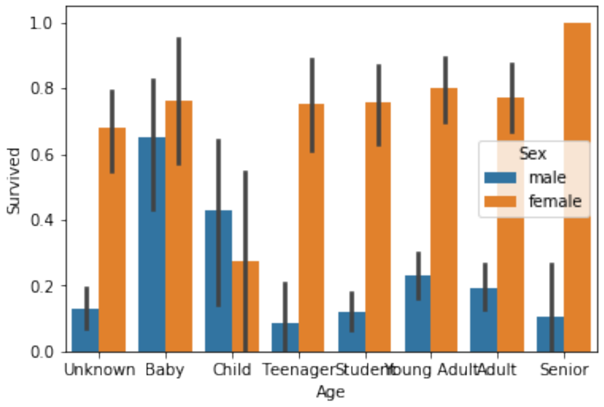


FIGURE 9

The graph illustrates the distribution of the fare variable with respect to the survived rate and we can say that people in 3rd quartile had more chance of survival than other groups.

```
'Unknown', '1_quartile', '2_quartile', '3_quartile', '4_quartile']
```

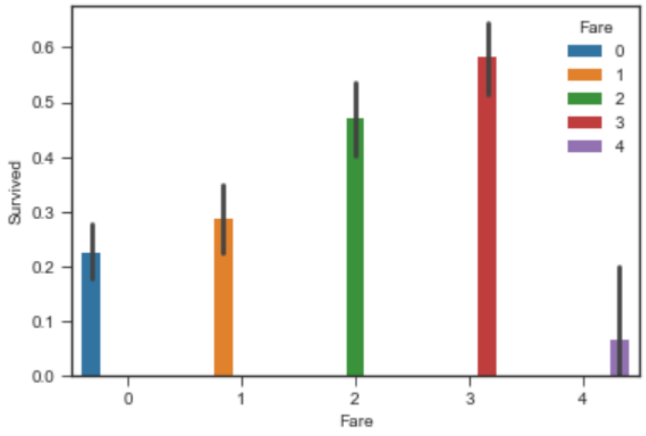


FIGURE 10

DATA HANDLING and FEATURE ENGINEERING:

FEATURE TRANSFORMATION:

- Aside from 'Sex', the 'Age' feature is second in importance. To avoid overfitting, each age value of passenger is grouped into logical human age groups. Passengers who age is between

[-1 & 0] - UNKNOWN
[0 & 5] - BABY
[6 & 12] - CHILD
[13 &19] - TEENAGER
[20 & 25] – STUDENT
[26 & 35] – YOUND ADULT
[36 & 60] – ADULT
[61-100] - SENIOR

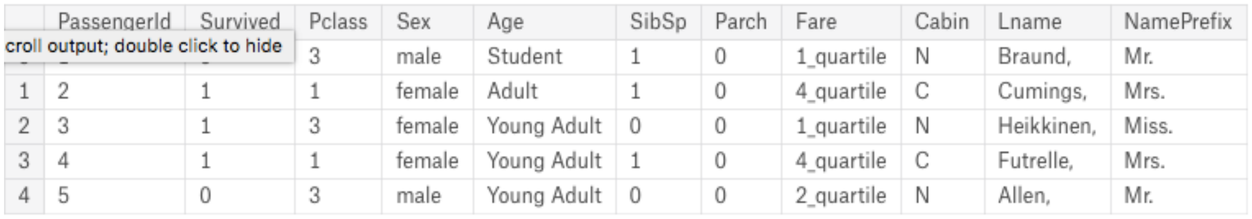
- As each ‘Cabin’ starts with a letter. It is assumed that the letter is more important than the number that follows, so, it was sliced off. Then all the null values (N/A) are replaced with ‘N’.

- Fare is another continuous value that was simplified. First, we replaced all the null values with -0.5. Then, we ran data_train.Fare.describe() to get the distribution of the feature, then placed them into quartile bins similar to ‘AGE’ variable.

[-1 & 0] - UNKNOWN
[0 & 8] - 1_quartile
[9 & 15] - 2_quartile
[16 & 31] - 3_quartile
[32 & 1000] - 4_quartile

- Extract information from the 'Name' feature. Rather than use the full name, I extracted the last name and name prefix (Mr. Mrs. Etc.), then appended them as their own features.
- Lastly, we dropped useless features. (Ticket, Embarked and Name).This was determined using correlation plots which showed that features are insignificant either because they have no variance or are dependent on other features.

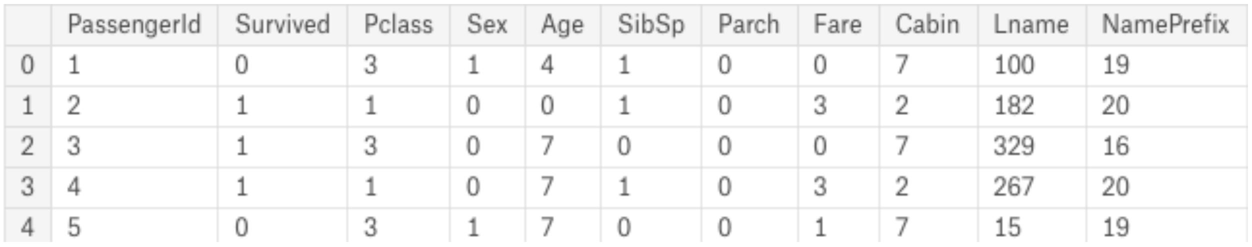
The dataset after the variables are feature engineered.


croll output; double click to hide

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Lname	NamePrefix
[illegible]	[illegible]	[illegible]	3	male	Student	1	0	1_quartile	N	Braund,	Mr.
1	2	1	1	female	Adult	1	0	4_quartile	C	Cumings,	Mrs.
2	3	1	3	female	Young Adult	0	0	1_quartile	N	Heikkinen,	Miss.
3	4	1	1	female	Young Adult	1	0	4_quartile	C	Futrelle,	Mrs.
4	5	0	3	male	Young Adult	0	0	2_quartile	N	Allen,	Mr.

*The code for the above *transform_features* functions are documented at the end of the report in appendix section.

As a next step, all other existing labeled featured such as ‘Sex’, ‘Age’, ‘Fare’ etc. are converted into numerical numbers by using *LabelEncoder* in Scikit Learn. By these preprocessing techniques, the data will be more flexible for analysis using different algorithms.



	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Lname	NamePrefix
0	1	0	3	1	4	1	0	0	7	100	19
1	2	1	1	0	0	1	0	3	2	182	20
2	3	1	3	0	7	0	0	0	7	329	16
3	4	1	1	0	7	1	0	3	2	267	20
4	5	0	3	1	7	0	0	1	7	15	19

*The code for the above *LabelEncoder* process was documented at the end of the report in appendix section

Scatterplot:

```
sns.pairplot(data_train,hue='Survived')
```

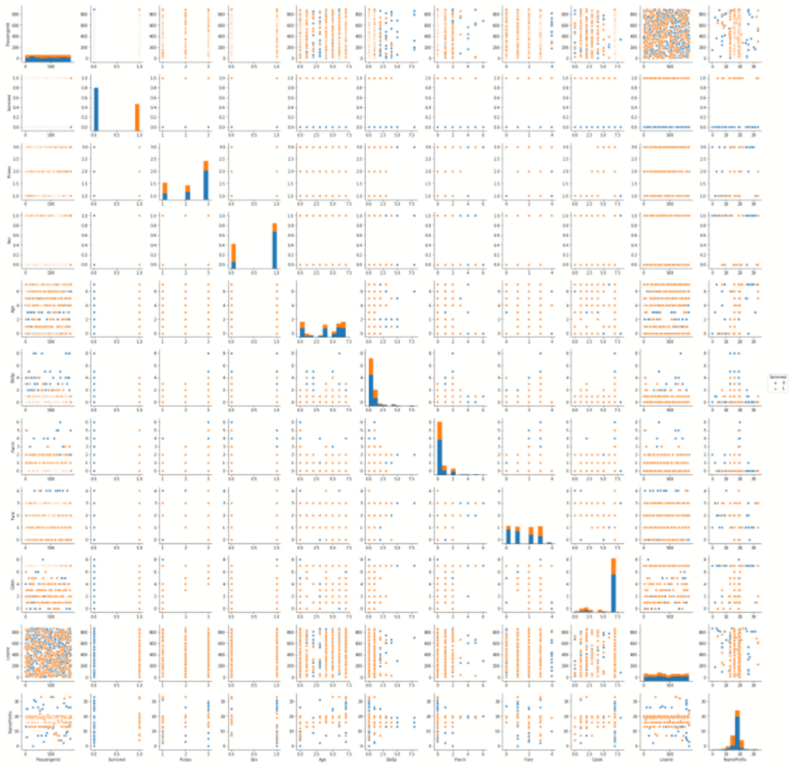


FIGURE 11

Pairplots illustrates that dependent variables “*Sibsp*” & *“Parch”* are positively skewed and “Cabin” is highly negatively skewed.

Correlation Plot:

```
t =
sns.heatmap(training_data[["Survived","Pclass","SibSp","Parch","Age","Fare"]]
.corr(),annot=True, fmt = ".3f", cmap = "coolwarm")
```

```
t1 = sns.heatmap(data_train[["Survived","Sex","Pclass","SibSp","Parch","Age",
"Fare","NamePrefix"]].corr(),annot=True, fmt = ".2f", cmap = "coolwarm")
```

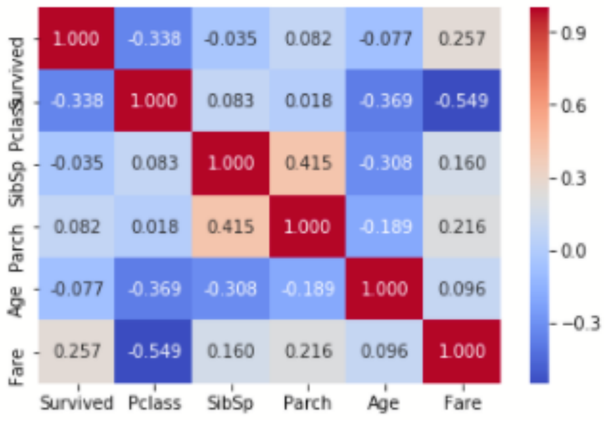


FIGURE 12

From the above plot, we can interpret that all the variables are independent of each other. So, we are considering all the variables as stated in the model.

So summarize:

Our Features are: PassengerID, Pclass, Sex, Age, SibSp, Parch, Fare, Cabin, Lname, NamePrefix

Prediction label: Survived(binary)

LANGUAGES USED:

Exploratory data analysis was done using R and python (using matplotlib). The classifiers were implemented using Python in Scikit-learn.

Github link of the code: https://github.com/vishaladdala/Machine-Learning-Project/blob/master/MachineLearningProject_Final.ipynb

MODEL BUILD:

From the EDA and Preliminary analysis, we concluded that the stated problem can be solved Machine Learning Classification techniques. The following are the classifiers that were implemented.

- Decision Tree
- Artificial Neural Network
- Deep Neural Network
- Support Vector Machine
- Multinomial Naïve Bayes
- Logistic Regression
- K-NN classifier
- Random Forest Classifier
- Adaboost
- Gradient Boosting Classifier
- Perceptron
- XGBoost Classifier
- Voting Classifier

In the process of Model Building, Prediction and Evaluation, we approached two methodologies: One without scaling and the other with scaling the variables. In the next sections, the methods of model building and evaluations are explained in an elaborated manner.

PHASE 1 (APPROACH 1(WITHOUT FEATURE-SCALING)):

In this approach, we ran the dataset through multiple classifier techniques and analyzed the change in accuracy with respect to parameter tweaking. We also implemented the technique of cross fold validation in each model to lower the variance of the model.

PHASE2(APPROACH 2 (WITH FEATURE-SCALING)):

In this approach, "Feature scaling" was applied to reduce the variance in the data. It aims to bring all the numerical values scaled down between 0 & 1 to reduce the effect of skewness by large numerical values. Generally in some of the classifiers, large numerical values are considered more weigh than others. To reduce this disparity, concept of 'Feature Scaling' is performed.

Fortunately, Classifiers such as Artificial Neural Networks, Deep Neural Networks, KNN, Perceptron classifiers accuracy was drastically improved with 'Feature Scaling'.

PERFORMANCE EVALUATION:

ACCURACY:

Accuracy is an proportion of correct classified instances with respect to the total instances. As it is one of the standard evaluation metric, we calculated the accuracy of all the classifier with respect to both the approaches. From the following table(Table 1), classifiers outstandingly improved with the scaling approach.

Classifier	Parameters	Approach 1 (Accuracy)	Approach 2 (Accuracy)
Decision Tree		0.760692488263	0.745356583948
Artificial Neural Network	*hidden_layer_sizes =(100,100), max_iter = 1000,alpha = 0.01, momentum = 0.7*	0.512438520009	0.791861725911
Deep Neural Network	hidden_layer_sizes =(100,100,100,100), max_iter = 100,alpha = 0.3, momentum = 0.7,activation = "relu"	0.531324055444	0.803170690812
Support Vector Machine	C = 100, kernel = "rbf"	0.618025933378	0.79598759222
Multinomial Naïve Bayes	alpha = 0.1	0.53073384753	0.73565448245
Logistic Regression	C = 10, max_iter = 10)	0.714366756092	0.79590878605
K-NN Classifier	n_neighbors= 5,leaf_size = 30	0.565828303152	0.779065504136
Random Forest Classifier	n_estimators = 100, max_depth = 10	0.829813324391	0.835388441762
AdaBoost	n_estimators = 10, learning_rate = 1	0.802991281019	0.802991281019
Gradient Boosting Classifier	n_estimators = 30,learning_rate = 1	0.829913369104	0.821363179074

Perceptron		0.55364017438	0.711349765258
XGBoost		0.815669572994	0.815669572994
Voting Classifier	`('dt', clf1), ('knn', clf2), ('svc', clf3)], voting='soft', weights=[2,1,2]`	0.787516208361	0.787516208361

TABLE 1

Out of all Random Forest Classifier and Gradient Boosting Classifier outstands others with the accuracy values ~83% & ~82% respectively in both approaches. But from the literature, it was known that accuracy value alone cannot be considered as the metric for efficiency of the model because of its known limitations. So, the other evaluation metrics were also performed as follows.

F1SCORE

F1 Score is another metric that considered as evaluation metric for verify the efficiency of the model. As F1 Score considers both precision and Recall, it's is considered as one of the good way to summarize the model evaluation. The Table 2 illustrates the F1 score of all classifiers.

Classifier	Approach 1 (F1 Score)	Approach 2 (F1 Score)
Decision Tree	0.753827967807	0.755076570534
Artificial Neural Network	0.53620780237	0.787595573441
Deep Neural Network	0.588330538788	0.78890509725
Support Vector Machine	0.618025933378	0.79598759222
Multinomial Naïve Bayes	0.53073384753	0.73565448245
Logistic Regression	0.714366756092	0.79590878605
K-NN Classifier	0.565828303152	0.779065504136
Random Forest Classifier	0.833960429242	0.821362061256
AdaBoost	0.802991281019	0.802991281019

Gradient Boosting Classifier	0.822851553767	0.824240442656
Perceptron	0.55364017438	0.711349765258
XGBoost	0.815669572994	0.815669572994
Voting Classifier	0.787516208361	0.787516208361

Out of all implemented classifiers, the following are the best performing classifiers:

1) *RandomForest:* Accuracy - 0.835388441762, F1 Score - 0.821362061256
2) *GradientBoostingClassifier*: Accuracy - 0.821363179074, F1 Score - 0.824240442656
3) *XGBoost:* Accuracy - 0.815669572994, F1 Score - 0.815669572994

HYPERPARAMETERTUNING:

As a next step, we tuned the hyper parameters of our best performing classifiers. For Random Forest Hyper parameter tuning, Grid Search Method was implemented and the results are

```
GridSearchCV took 95.87 seconds for 768 candidate parameter sett
ings. Model with rank: 1 Mean validation score: 0.829 (std: 0.02
7) Parameters: {'bootstrap': False, 'criterion': 'gini', 'max_de
pth': 100, 'max_features': 3, 'min_samples_leaf': 3, 'min_sample
s_split': 10}
```

For XG Boost Classifier Hyper Parameter Tuning:

```
GridSearchCV took 68.42 seconds for 1536 candidate parameter
settings. Model with rank: 1 Mean validation score: 0.815 (std:
0.037) Parameters: {'colsample_bytree': 0.7, 'learning_rate':
0.05, 'max_depth': 6, 'min_child_weight': 5, 'missing': -999,
'n_estimators': 100, 'nthread': 1, 'objective':
'binary:logistic', 'seed': 27, 'silent': 1, 'subsample': 0.8}
```

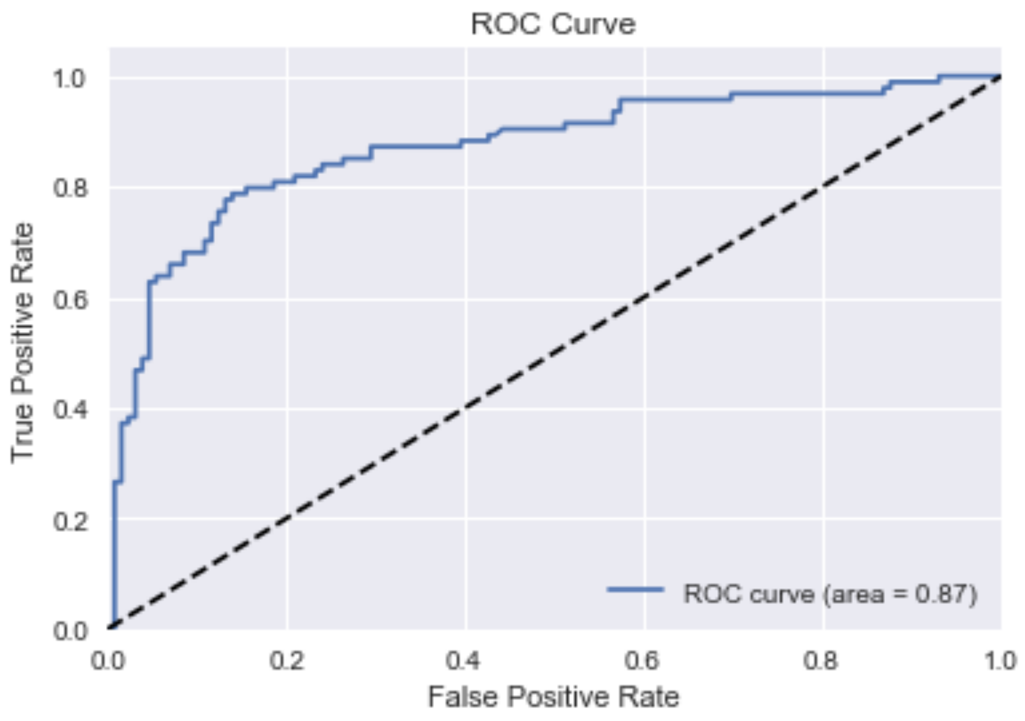
For Gradient Boosting Classifier Hyper Parameter Tuning:

```
Model with rank: 1 Mean validation score: 0.830 (std: 0.011)
Parameters: {'max_depth': 14, 'max_features': 7,
'min_samples_leaf': 20, 'min_samples_split': 100, 'subsample':
0.8}
```

ROC Curve

This metric is also widely used across machine learning algorithms, which considers True positive verses False positive. The closer the curve to the upper left corner, the better the classifier's performance is.

```
ROC AUC: 0.87
```



ROC CURVE FOR XG BOOST CLASSIFIER

FIGURE 13

ROC CURVE FOR RANDOM FOREST CLASSIFIER

```
ROC AUC: 0.87
```

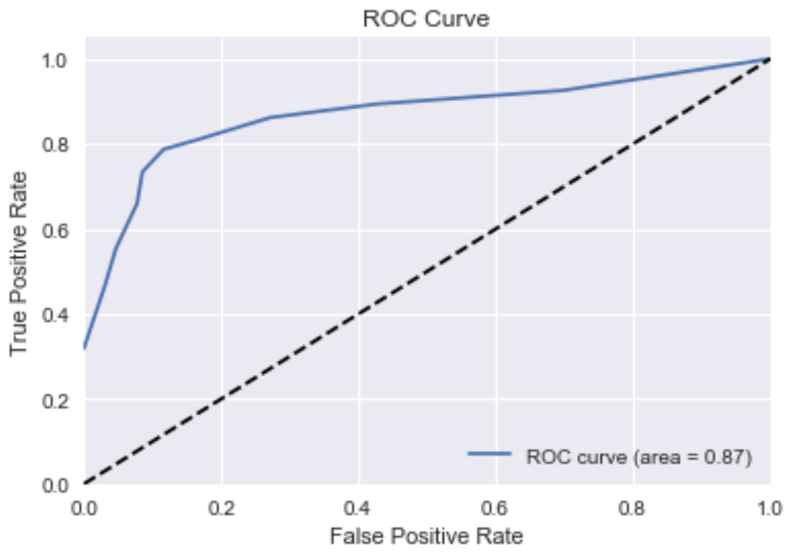
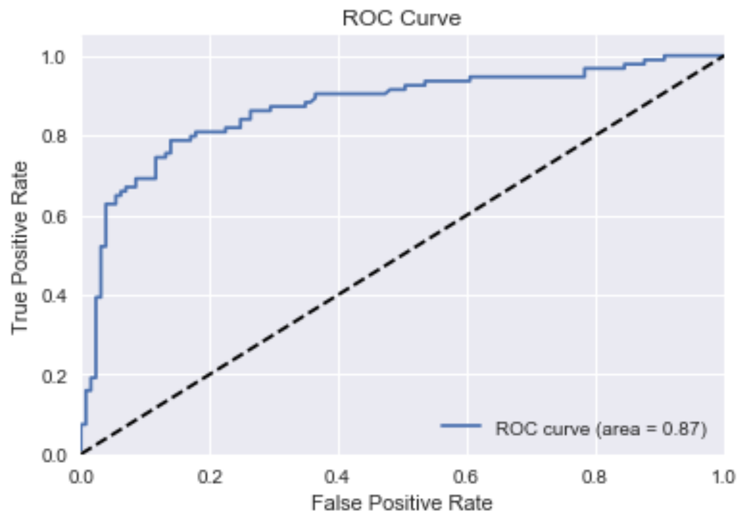


FIGURE 14

ROC CURVE FOR GRADIENT BOOST CLASSIFIER

```
ROC AUC: 0.87
```



BEST 3 CLASSIFIERS FOR OUR DATA:

The Following are the best classifiers that predicts more accurately, with highest accuracy score, F1 Score and ROC curve area.

Xgboost

We implemented XGBOOST to obtain accurate prediction on binary classification by fitting the resulted boosted trees. It is an optimized booster algorithm with power, speed in terms of computation during training phase and better application of regularization. Training the model with xgboost had escalated our efforts towards producing almost correct results. This classifier heavily depends on the parameter: Number of trees. Hence, this parameter is considered the hyper-parameter for this algorithm.

We tried initially, with a smaller number of trees, checked results and then increased that number accordingly when we realized a degrading performance and hence trained six number of trees.

With this procedure, we observed an accuracy of 81.56% on our data classification.

Random forest

This classifier is the simplest implementation that can be used to quickly train a model. Number of trees used in the implementation is the hyper-parameter. It has been inferred that even if this number increases, this classifier does not overfit the model since selection is performed randomly. It is during training time that these weaker decision trees are grown simultaneously and combined to form a stronger random forest. For tuning the hyper-parameters, GridSearch technique has been used in combination with Random Forest Classifier. With this procedure, we observed an accuracy of 82.13% on our data classification.

Gradient Boosting

This is one of our best performing classifiers since it reduced the loss while resulting in a predicted distribution of data point probability almost equal to that of true distribution. We observed this true nature of gradient boosting when we applied the default 'deviance' or 'logistic regression' function as the loss function value and performed 100 stages of boosting to obtain best results of probabilities thereby, predictions. We observed that tuning of parameters particular to trees, which are the learners that are combined in this procedure, improve the performance of the classifier and for the purpose of tuning the hyper-parameters, Grid Search technique has been used in combination with Gradient Boosting Classifier. With this procedure, we observed an accuracy of 82.42% on our data classification.

CONCLUSIONS:

We did the project in two phases. In the first phase although we did preprocess our data, some of our classifiers using gradient descent technique like neural networks, deep neural networks and classifiers like support vector machines, Naïve Bayes, Perceptron, k-NN classifier were not performing well on our data. However, ensemble methods such as Random Forests, AdaBoost ,Gradient Boosting Classifier were performing well and giving good accuracies.

Therefore, in phase 2 of our project we decided to use the concept of feature scaling to improve the performance of all our classifiers. After Feature scaling, the performance of neural network, deep neural network, support vector machines, Naïve Bayes, Perceptron, k-NN classifier improved drastically. Also, there was a slight increase in the performance of our ensemble methods.

Finally, we shortlisted our best 3 classifiers to be Random Forest, XGBoost, and Gradient Boosting Classifiers. We then proceeded to perform hyperparameter tuning for these 3 classifiers using grid search and tried to find the best set of hyperparameters for these classifiers to make their performance on our dataset the best as possible. We also evaluated our best models using the ROC Curves.

To summarize, Random Forest, XGBoost, and Gradient Boosting all perform almost equally well on our dataset. But if we had to pick one model we would pick the "Random Forest Classifier" as our best model which gave us the best precision of 83.11% and has an ROC AUC of 0.87.

REFERENCES:

http://scikit-learn.org/stable/supervised_learning.html#supervised-learning