Application of machine learning algorithms to predict the survival outcome titanic Dataset

Michael Ajao-Olarinoye
Faculty of Engineering, Environment and Computing
MSc Data Science and Computational Intelligence
Coventry university
Coventry, United Kingdom
Olarinoyem@uni.coventry.ac.uk

***Abstract*—This paper looks into the techniques used in preprocessing, feature engineering, machine learning evaluation, to predict the binary classification of the titanic dataset using Logistical Regression, Support Vector Machine, Random Forest, Stochastic Gradient Descent (SGD), K Nearest Neighbor, Gaussian Naive Bayes, Decision Tree. The algorithm was run on a jupyter notebook using python and Scikit-learn library. This dataset was used as a competition on Kaggle.**

I. INTRODUCTION

The tragedy that happen on April 15 1912, to a British passenger liner after hitting an iceberg during its voyage from Southampton to New York where more than 1500 people died is one of the world's deadliest commercial marine disaster. The ship name RMS Titanic was said to be carrying an estimate of 2,224 people both crew and passengers according to Wikipedia (Wikipedia, 2020). The ship was designed to have safety features but still was said to have been carrying 20 lifeboats which could only contain 1178 people. In spite of the fact that there was some component of karma associated with enduring the sinking of the ship, a few gatherings of individuals were bound to get by than others, for example, ladies, youngsters, and the privileged (upper-class). The RMS Titanic was the biggest ship on the sea at the time it was built and was the second of three Olympic-class sea liners worked on by the White Star Line. The Titanic was created by the Harland and Wolff shipyard in Belfast. Thomas Andrews, her designer, passed on in the calamity (Donges, 2018).

The wreck area of the liner Titanic has been found by a joint U.S-French campaign off Canada in waters, in excess of 13,000 feet below, the declaration didn't give the exact area of the disaster area, evidently to look after security (services, 1985). The ship called unsinkable which has been an inspiration for fiction and non-fiction books, several survivors wrote about their experiences but the first historical book to be publish about the experience that occurred during the sinking was named *A Night to Remember* (Walters, 1955). In this paper I'll be discussing and comparing different classification and regression machine learning algorithms to the titanic dataset and my process used in this paper is as follows

- II describes the problem involved in the dataset and the type of prediction(evaluation) we want to achieve
- III covers the machine learning algorithm to use
- IV talks about the preprocessing and feature engineering involve in the modelling of our data
- V gives a brief discussion on the conclusion derived from the hyper-parameter tuning processed performed on the dataset
- VI shows the results metrics gotten from the model experimentation performed on the dataset

II. PROBLEM DEFINATION AND DATASET

A. *Problem definition*

The dataset was gotten from Kaggle, a competition created for people to show case their machine learning skills and to show and build different machine learning models or algorithm that predicts the survival of people on the day of the accident. Right now, solicited to finish the examination from what sorts of travelers were probably going to survive the catastrophe utilizing Machine learning. So, we must foresee if a traveler made due from the sinking Titanic or not with the assistance of Machine learning.

B. *Dataset*

The chief hotspot for information about Titanic travellers is the Encyclopedia Titania. The datasets utilized here were started by an assortment of specialists. One of the sources is Eaton and Haas (1994) Titanic: Triumph and Tragedy, Patrick Stephens Ltd, which incorporates a traveller list made by numerous analysts and altered by Michael A. Findlay. The data set has been split for us to use into two which are:

1. *Train.csv*
2. *Test.csv*

The train dataset has a total of 891 input rows and 11 features with the target variable (survived) while the test dataset has a

total of 418 input rows values and not the target variable (survived).

Table I. Dataset Dictionary

Variable	Definition	Key	Type
PassengerId	Unique identifier # for each passenger		Numeric
Name	Name of passenger with title		Text Variable
survival	Survival outcome	0 = No, 1 = Yes	Numeric
Pclass	An intermediary for financial status	1 = 1st(upper) 2 = 2nd(middle) 3 = 3rd(lower)	categorical
sex	Gender specification	Female Male	categorical
Age	Age in years		Numeric
Sibsp	# of siblings / spouses aboard the Titanic		Numeric
parch	# of parents / children aboard the Titanic		Numeric
ticket	Ticket number		Text Variable
fare	Passenger fare		Numeric
cabin	Cabin number		categorical
embarked	Port of Embarkation	C = Cherbourg Q = Queenstown S = Southampton	categorical

III. MACHINE LEARNING MODELS IMPLEMENTED

Since the problem we are trying to solve in the dataset is a classification problem which is predicting the survival of the people on board the titanic, this paper discusses about using the following classification algorithm to compare which predicts best for this supervised learning dataset:

- Logistical Regression
- Support Vector Machine
- Random Forest
- Stochastic Gradient Descent (SGD)
- K Nearest Neighbor
- Gaussian Naive Bayes
- Decision Tree

A. Logistical regression

Logistic regression is in fact a commonly used supervised classification method. Logistic regression and its extensions, such as multinomial logistic regression, allow us to predict the likelihood that an observation of a certain class using a simple, well-understood approach (albon, 2018). Logistic regression is a widely used binary classifier, despite having regression in its name (i.e. the target vector can only take two values positive class "1" of negative class "0"). Logistic Regression uses a more complex cost function, which can be described as 'Sigmoid' or 'logistic function' instead of a linear function.

$$f(X) = \frac{1}{1+e^{-(x)}} \quad (1)$$

$$Z = \beta_0 + \beta_1 X \quad (2)$$

$$h\theta(X) = sigmoid(Z) \quad (3)$$

$$h\theta(X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}} \quad (4)$$

equation(1) defines the sigmoid function formula, equation(2) is an hypothesis of a linear regression formula, equation(3) and (4) shows the equation that defines the hypothesis of a logistic regression (pant, 2019). We will use the Logistic Regression to verify our priorities making and completing function expectations and decisions. That can be achieved by measuring the signature coefficient in the decision function.

B. Support Vector Machine

Support vector machines (SVM) are powerful algorithm for classification that also aims to find the best possible boundary between two created groups. SVM Operate by finding a judgement method that aims to separate the training data into two groups. The decision function is chosen to optimize its distance to the closest training data on either side of the board. If no decision function is capable of splitting the data linearly, the transformation function of the kernel is used to map the data to another dimensional space (called a function space) such that it can be linearly segregated using normal vector support system decision-making techniques (Boyle, 2011). A kernel is merely a function that takes two input data points and returns a score of similarities. The similarity can be viewed as a proximity metric. The closer the data points are, the greater the resemblance.

C. Random Forest

Random forests are a mixture of tree predictors, such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest (Breiman 2001). A random forest is a collection of decision trees based on random samples with a different strategy for dividing a node (Bonaccorso 2017). Random Forest Models, with a minor tweak, can be thought of as Bagging. When determining where to decide and how to make choices, Bagged Decision Trees shall have the maximum range of

features to choose from. Random Forest models determine where to break on the basis of a random set of features. Rather than breaking up at similar features at each node throughout, Random Forest models have a distinction degree as each tree can split based on different features (Lutins, 2017).

D. Stochastic Gradient Descent (SGD)

SGD which implements a very popular algorithm that can be applied to a variety of different loss or error functions. To talk about stochastic gradient descent, we have to talk about gradient descent which is a method to optimized a linear model. To fit the line to the data the error is reduced by using gradient descent (Shalev-Shwartz and Ben-David 2014). By taking the gradient of the error functions against the weight (w_i) taking a step in the direction of the negative of the gradient till the error function is decreased to its minimum.

$$w_i \rightarrow w_i - \alpha \frac{\partial}{\partial_{w_i}} Error$$

Instead of using the whole dataset a few samples are selected at random to run each iteration, the problem of having a large dataset and performing gradient descent to use the whole dataset to get to the minima in a less random or noisy way is solved by the batch system employed in gradient descent which uses only one sample, i.e. one batch size, to perform each iteration. The sample is randomly shuffled and chosen for iteration.

E. K Nearest Neighbor

This algorithm is a simple to implement supervised learning algorithm. KNN operates by calculating the distances between the query and all the examples in the results, choosing the number of examples (K) nearest to the query, then voting for the most frequent label. But it has a significant downside of being increasingly sluggish as the scale of the data increases in use (Harrison, 2018).

F. Gaussian Naive Bayes

We need to understand Naïve Bayes to be able to figure out how Gaussian Bayes works. Naive Bayes is a binary (two-class) classification algorithm and multi-class classification problems. Using Bayes theorem, we will consider the probability that A will happen, provided that B has happened Naïve Bayes equation give below

$$P(A/B) = \frac{P(B/A)P(A)}{P(B)}$$

The premise here is that the predictors / features are distinct. This is the appearance of one particular attribute does not affect the other. Therefore, it's called naïve. Unless the predictors have a constant value and are not discrete, instead we conclude that such values are derived from a gaussian distribution.

G. Decision Tree

Decision Tree algorithm is part of the supervised learning algorithms family. Like other supervised learning algorithms, algorithms for decision tree can also be used to solve regression and classification problems. The general rationale for using Decision Tree is to create a training model that can be used to predict the class or value of target variables by learning decision rules inferred from prior data (training data). The decision tree algorithm is attempting to solve this problem by using tree representation. -- internal tree node corresponds to the descriptor, and each leaf node corresponds to the symbol of the class (Saxena, 2017).

IV. PREPROCESSING AND FEATURE ENGINEERING

In this paper I used scikit learn library to perform the exploratory analysis and machine learning process involved using jupyter notebook. The dataset which was gotten from Kaggle. Preprocessing started by loading the dataset which is in csv form into data frame, two different data were loaded which is the train.csv and the test.csv. there are several process that is going to take place in this paper, data exploration to see how the data given might tell a story or describe a problem or solution, the visualization of the data, statistical analysis, creating new features from existing features, converting various data into categorical data for the learning process and working with missing data .

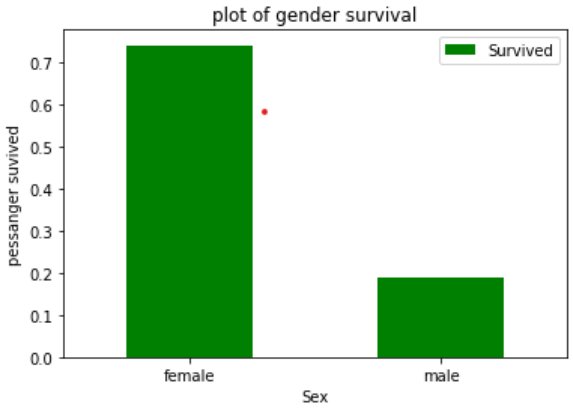

Figure I. The plot showing the rate of people that survived according to gender

From the preliminary investigation into the dataset, it was seen that the survival of women was far greater than that of men, and to also discuss the facts that women and children are always the first to be protected in the occurrence of disasters. In the paper, the effects of family, gender and age are going to be used as features to draw conclusion on the learning process. Features such as family size was created by adding the parch feature and the sibsp feature to determine the sizes of families on the ship and how it had on effect on the survival analysis.

For the continuation of the preprocessing and analysis I decided to concatenate the test and train data frame together to create a data frame of 1309 rows called titanic_df and assigned the null values generated in the survived feature “null” values. It is convenient to work with concatenated data for this missing data problem. The data had a total of 4 features columns with missing data. Dealing with missing data in the data frame, the

Cabin column has 1014 of missing data, the Age column with 263 of missing data, Embark column with 2 missing data and the Fare column with 1 missing data.

The features call Cabin has 1309 data missing there were two choices to be made which was to either get rid of the feature or to find an appropriate way to use them. The amount missing is more than 70% of the data required, after research and deliberation I decided to drop the entire feature as I feel it won't be beneficial to the models. Other works that I found and researched decided to work with segmenting the socio-class of people in the ship.

The other features with missing data Age, Embark and fare will be filled with statistical descriptive data. More analysis was done on the feature Embarked which showed that it's a categorical feature with 3 values C, Q and S which was the port that people embarked from onto the ship. It was shown that 70% of the people embarked from Southampton so the two missing value I decided to replace them with S which is Southampton. The Fare feature also ad one missing data, since it was numeric, I decided to use the mean of all the fare features to replace the particular missing data.

Before I talk about the Age feature missing data, I'll talk about a few new features created from the existing once. From the names features I created an initial features which hold 4 categories[master, master, Mr., Mrs.] other title types, like capt, col, don, countess, lady, major etc. that were in the data were crosstab with the sex feature to determine which sex there were and imputed, which resulted in the data imputed in the right place. The graph below shows the median age of the initial features which we will use to determine the values to be imputed into the missing data age feature.

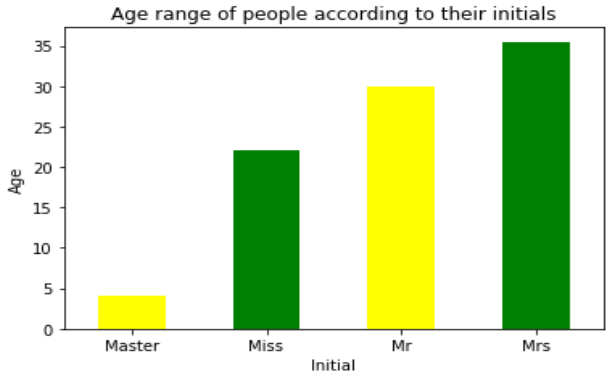

Figure II. Age range of people according to the initial feature

The age feature was resolved my assigning to the location the median of each initial to the required age. Three different features were determined by me not to be useful to this context. Name, ticket and passengerId features were the features dropped passengerId was drop as not to be having an effect on the target features, the ticket feature could have been a good feature to use but as it was a text variable and not numerically organized, and with the research I did, I could not determine the correct format for the feature, so I decided to drop it. While the name feature was dropped a new feature was generated from it, that would help the models to predict better.

The following features were dropped (Parch, Sibsp) as a result of creating another feature called family size which groups the type of size of family as a categorical value which are (single, small family, medium family, large family) to check if the family size affected the survival rate. The Fare was full of irregular data which was skewed, even if it was scaled could lead to the overweight of high value in the model. I'll reduce the skewedness of the feature by transforming with a log function.

The data frame present at the point was good to run our model, but had a little bit of change that needed to be applied to it, which include converting some feature columns from categorical data. In this process I implemented the get_dummies pandas function provided, get_dummies convert the data applied into zeros and ones creating new columns. The features that the dummy function was applied to are Pclass, Sex, initial and embark. Now we have a data frame to implement machine learning algorithms on, the correlations of the resulting features were taken and shown in the figure below.

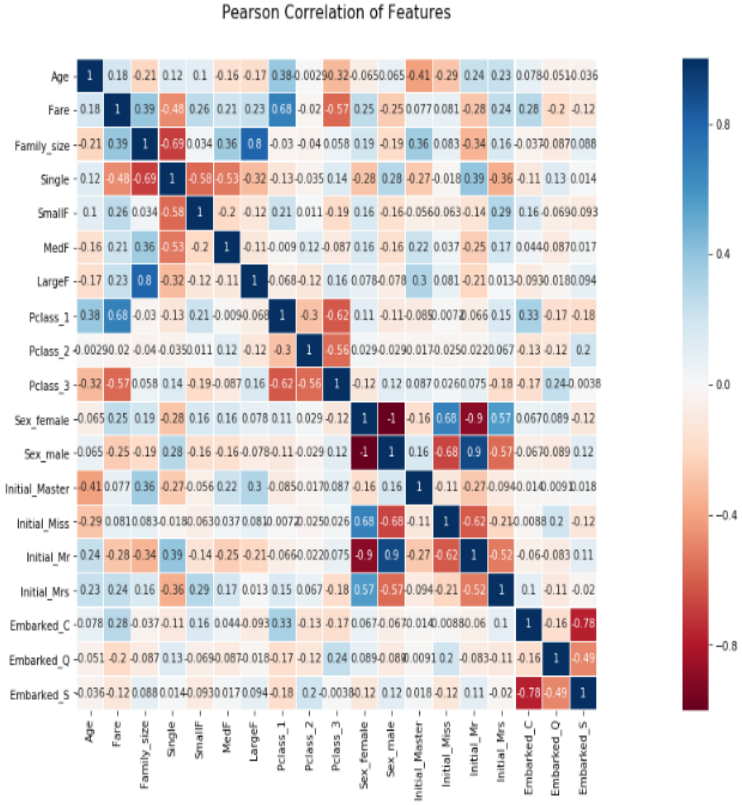

Figure III. Correlation plot of all features

V. MODEL HYPERPARAMETER TUNING

The data frame titanic_df which was concatenated at the beginning of the preprocessing was split back into the train and test data frame with both been free from missing data train and test data frame shape respectively (891, 20), (418, 19). I'll be looking for the goldilocks model, one which fits well on our dataset and also well on unseen examples. Since all the algorithm in scikit-learn library implements the same model.fit() function and model.score() function which return

the ration of predictiveness 1.0=100%. we would evaluate the chosen algorithms to see how they perform without turning the hyper-parameters using the train_test_split function provided to us by scikit-learn library to spilt the train data frame into (X_train, X_test, y_train, y_test) using 20% as the test size from the train data frame.

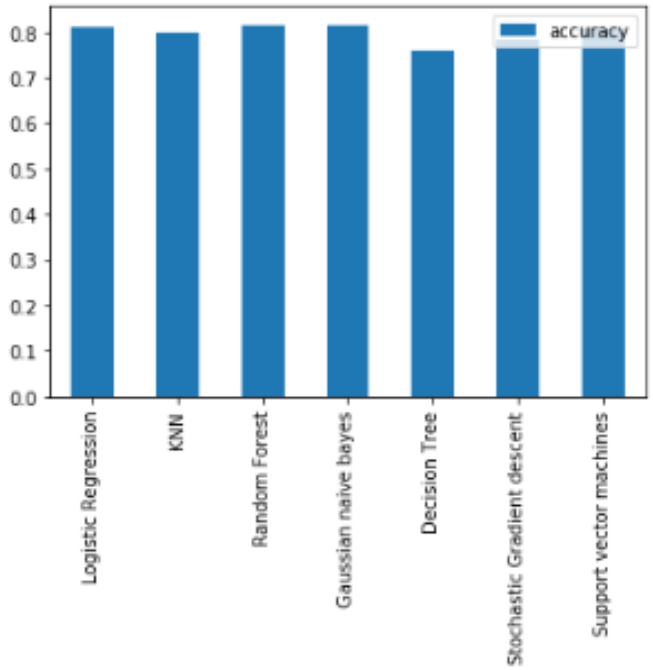

Figure IV. Bar plot of the accuracy of the trained model

From the evaluation performed we could see that logistic regression, support vector machines, decision tree, Stochastic Gradient descent and random forest performed approximately 80%, with Logistic regression and gaussian naive bayes performing best at approximately 82% and decision tree coming as the lowest with approximately 76%. Since this is our base line model first prediction, with most of the algorithms showing values ranging from approximately 76% to 82%, I'll used various method of hyper parameter turning to improve the models.

Table II. Table of the trained model

Trained Model	**Accuracy**
Logistic regression	81%
KNN	80%
Random forest classifier	82%
Gaussian naïve bayes	82%
Decision tree	76%
Stochastic Gradient descent	78%
Support vector machines	80%

Hyperparameter tuning by hand

From research I found out that you could perform tuning by changing the values of some hyperparameters of the algorithm, so I tried experimenting with the KNN algorithm that produced an evaluation of approximately 80% to see if it can be improved better, n_neighbors which is an important parameter is the number of neighbors used for the kneighbors queries by default the values is an interger of 5, so looping through a range of neighbor values from range of 1 – 20 to see if we can improve the algorithm by changing this one parameter. The graph below shows that at when n_neigbors = 14, It actually improved and the maximum score on the training set was 81.01%

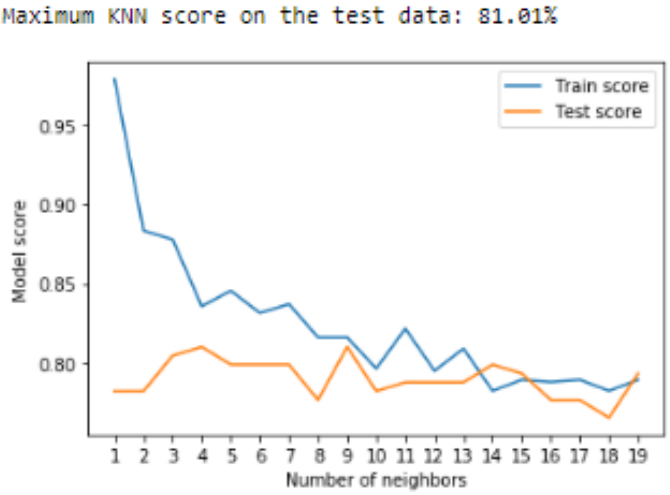

Figure V. Graph of KNN train score against test score

With this result we can determining that changing one parameter by hand has an effect over the performance of an algorithm, though it improved to 81.01% we can't go ahead with using KNN to solve the problem because there are other models that are performing better.

I tried tuning the RandomForestClassifier by hand by adjusting the number of n_estimators to 200 because I got a warning that it would be changed from 10 to 100 in a new version as the default. Since it's a parameter I tested it and got a slight improvement.

Table III. Performance of Random forest classifier with a change in hyperparameter

	Baseline	Baseline estimator = 200
Accuracy %	80.45	82.68
Precision %	76.71	79.73
Recall %	75.68	79.73
F1 %	76.19	79.73

Hyper parameter tuning with RandomizedSearchCV

Working with scikit-learn RandomizedSearchCV, I will use it to search over a grid the number of iterations across different chosen parameter to see which combination works best and save it. I'll be performing RandomizedSearchCV on but my logistic regression model, support vector machine and random forest to see the best search parameter that would be use to optimize the model. This part of the experiment was done using the train_test_split for the data. Taking performance into account the three was ran at n_jobs been at default means running the process on the default number of cores the processor has and it fit with cross-validation = 10 folds and number of iterations = 30 and other randomsearchCV

parameter. On running the code and tuning the hyper-parameters on logistic regression with the search parameter found to be best at {multi-class: auto, penalty: L2, C: 0.55} there was an increase in precision and decrease in both f1 score and recall and it took 2.3secs to run.

randomsearchCV performed on the randomforest classifiers took 3.3mins to run. The best parameter found was that at the point when {n_estimators: 910, min_samples_split: 18, min_samples_leaf: 1, max_features: auto, max_depth: 10} would produce the result found in the table below. Which saw an increase in precision but a reduction in accuracy, recall and f1 which are metrics to measure the performance of a model. I tried evaluating support vector machines with the randomsearchCV but it took a long time, so I decided to abandon the approach and move to gridsearchCV

Table IV. RandomSearchCV performance table

	Rs Random Forest	**Rs Logistic Regression**
Accuracy %	82.12	80.01
Precision %	81.82	78.57
Recall %	72.97	74.32
F1 %	77.14	76.39

Hyper parameter tuning with GridSearchCV

The difference between gridsearchCV and randomSearchCV is that it would run an extensive search on every possible combination and save the best found in the range of parameters given to it. The grid search was used on three different algorithms to try and see which performs best. KNN didn't perform that well when I did grid search on it. Logistic regression which is not a classification algorithm still perform quite close to the randomsearchCV with a reduction in recall. The random forest classifier on the other hand had an improvement in its metrics with its search parameter been at max_dept : 5 and n_estimator : 360 but for the rest of the experiment I'll be using random forest classifier with a parameter of n_estmators = 200 as this performed best in all my models built, I'll also be comparing the evaluation metrics of random forest at (max_dept : 5 and n_estimator : 360) which was the grid search result gotten, Gaussian naive Bayes out of the box without tuning and also the random search of logistic regression.

VI. MODEL EVALUATION AND CONCLUSION

I'll be using the following metrics to evaluate the chosen models.

- Precision
- Recall
- F1 score
- Confusion matrix
- ROC curve and AUC
- AUC area under the curve
- Classification report

a. Confusion matrix
This gives us an output matrix that shows us the complete performance of the model on the binary classification problem. The 4 main terms in confusion matrix are below:

i. TP – True Positive:
When YES is predicted and its actual output is YES

ii. TN – True Negative:
When NO is predicted and the actual output is NO

iii. FP – False Positive:
When YES is predicted and the actual output is NO

iv. FN – False Negative:
When NO is predicted and the actual output is YES

b. Accuracy
This is mostly referred to as mean, it tells us the proportion of observation we correctly labeled. Accuracy is not the only metric to use for measurement as some data maybe imbalance

$$\text{Accuracy} = \frac{TP+\text{TN}}{TP+FP+TN+FN}$$

c. Precision
Precision depends on the "positive" values predicted in the dataset. Through calculating the values based on precision, you decide whether you are doing a decent job of predicting the positive values relative to predicting the false negative values.

$$precision = \frac{TP}{TP + FP}$$

d. Recall
In your sample, Recall focuses on the real "positive" values. By optimizing recall values, you decide if you're doing a decent job of predicting positive values, regardless of how you're doing with the actual negative values.

$$recall = \frac{TP}{TP + \text{FN}}$$

e. F1 score
F1 Value is the harmonic mean of Recall and precision. The range is [0, 1] for F1 Quality. It shows you how accurate the classifier is (how many instances it correctly classifies), and how reliable it is (a large number of instances are not missing).

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

f. ROC curve and AUC

We can calculate the region under the curve by identifying various thresholds for our classification metrics. Similar to any of the other metrics above, this implies that when the AUC is higher (closer to 1), our model output is stronger than when our metric is below 0.

g. Classification report

Scikit-learn has a built-in function called classification report that returns some of the main classification metrics including precision, recall and f1-score

Random forest classifier (n_estimator = 200)

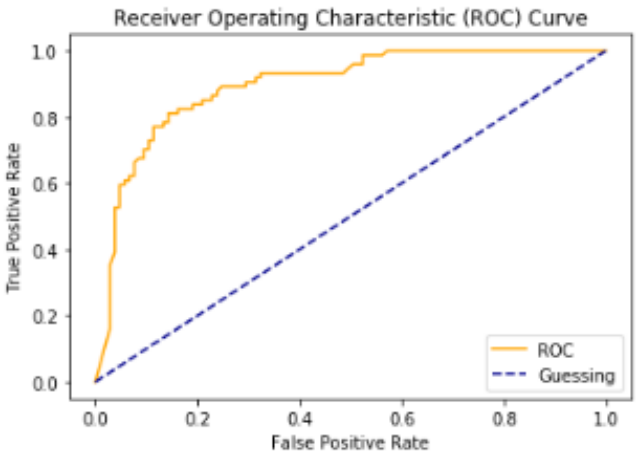

Figure VI. ROC curve for Random forest classifier (n_estimator = 200)

The AUC for this model was 0.90

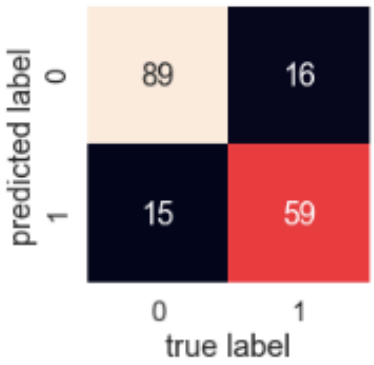

Figure VII. Confusion matrix for Random forest classifier (n_estimator = 200)

	precision	recall	f1-score	support
0	0.86	0.85	0.85	105
1	0.79	0.80	0.79	74
accuracy			0.83	179
macro avg	0.82	0.82	0.82	179
weighted avg	0.83	0.83	0.83	179

Figure VIII. Classification report for the Random forest classifier (n_estimator = 200)

The classification report for this model shows a good metric, that could predict the positive value correctly close to 80%. The model was also run on the original data that hasn't been split yet with cross validation at 5 folds and the measuring metrics were decided and plotted below, showing a reducing in the precision and other metrics

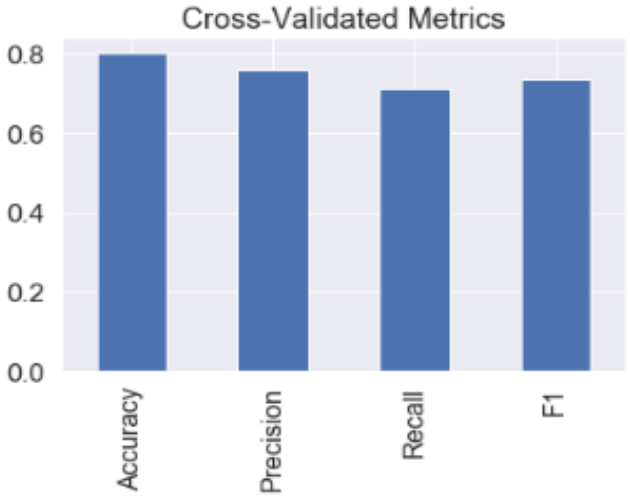

Figure IX. Bar plot of the cross validated metrics of Random forest classifier (n_estimator = 200)

Random forest classifier (max_dept = 5 and n_estimator = 360)

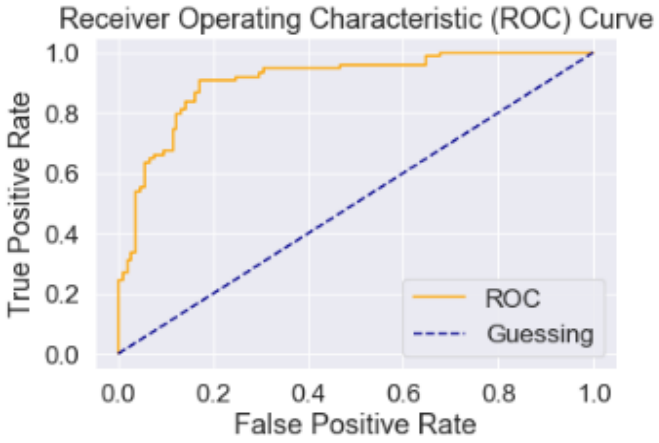

Figure X. ROC curve for Random forest classifier (max_dept = 5 and n_estimator = 360)

The AUC for this model was 0.90

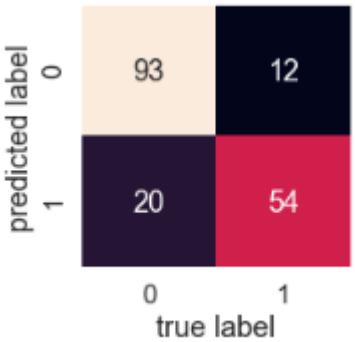

Figure XI. Confusion matrix for Random forest classifier (max_dept = 5 and n_estimator = 360)

Gaussian naive Bayes

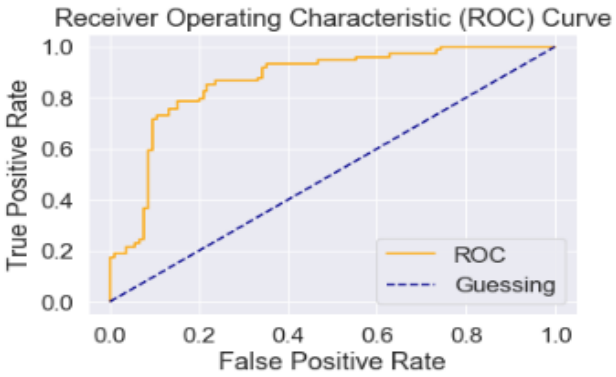

Figure XII. ROC curve for Gaussian naïve bayes model

The AUC for this model was 0.86

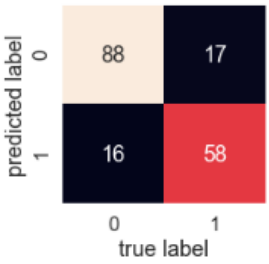

Figure XIII. Confusion matrix for Gaussian naïve bayes model

Logistic regression (penalty = l2, muilt-class = auto, c = 0.55)

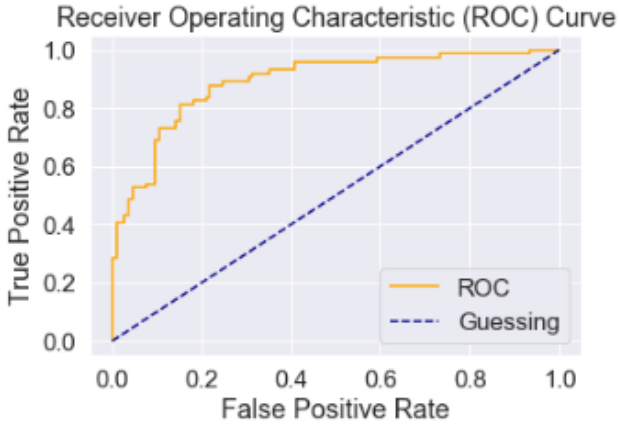

Figure XIV. ROC curve for logistic regression (penalty = 12, multi-class = auto, c = 0.55)

The AUC for this model was 0.89

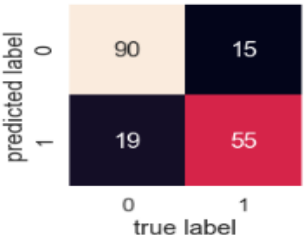

Figure XV. Confusion matrix for logistic regression (penalty = 12, multi-class = auto, c = 0.55)

CONCLUSION

The Random Forest Classifier with n_estimator = 200 was chosen as my best model to submit at this time as I decided on using the AUC metric and f1 score, the Area under the curve was approximately 0.90 and f1 score at approximately 0.80. this is the model I have decided to use for the prediction of the people that could have survive the horrible disaster, as we have seen in the data the people likely to survive were the women and the kids on the ship. In this report most classification algorithm used produced a good accuracy score from just out of the box without tuning. After research I found out that other people build models that performed better than this with better feature engineering and better search parameter.

During the experimentation I checked the feature importance to see which of the features where important in the model learning process and which wasn't important. Fare and Age feature were the top two features that aided in the learning process while Embarked_Q, small family and single features where not import to the learning process. Feature works include using ensembled method

VII. APPENDIX

I uploaded my data and my notebook file on git hub https://github.com/michaelajao/machine_learning

REFERENCES

albon, C., 2018. *Python machine learning cookbook.* 1st ed. califonia: O'Reilly media.

Boyle, B. H., 2011. *Support Vector Machines: Data Analysis, Machine Learning and Applications.* New york: Nova Science Publishers, Incorporated.

Donges, N., 2018. *Predicting the Survival of Titanic Passengers.* [Online] Available at: https://towardsdatascience.com/predicting-the-survival-of-titanic-passengers-30870ccc7e8
[Accessed 27 march 2020].

Harrison, O., 2018. *Machine Learning Basics with the K-Nearest Neighbors Algorithm.* [Online] Available at: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761
[Accessed 8 april 2020].

Lutins, E., 2017. *Ensemble Methods in Machine Learning: What are They and Why Use Them?.* [Online]

Available at: https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f
[Accessed 7 april 2020].

pant, A., 2019. *towards data science.* [Online] Available at: https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148
[Accessed 29 march 2020].

Saxena, R., 2017. *Dataaspirant.* [Online] Available at: https://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-works/
[Accessed 9 apirl 2020].

services, T. w., 1985. *Los angeles times.* [Online] Available at: https://www.latimes.com/archives/la-xpm-1985-09-03-mn-24236-story.html
[Accessed 25 march 2020].

Walters, L., 1955. *A night to remember.* new york: St. Martin's Griffin.

Wikipedia, 2020. *RMS Titanic.* [Online] Available at: https://en.wikipedia.org/wiki/RMS_Titanic
[Accessed 22 March 2020].

Breiman, L. (2001) 'Random Forests'. *Machine Learning* 45 (1), 5-32

Bonaccorso, G. (2017) *Machine Learning Algorithms* [online]. Birmingham: Packt Publishing, Limited. Available from <http://ebookcentral.proquest.com/lib/coventry/detail.action?docID=4926962>

Shalev-Shwartz, S. and Ben-David, S. (2014) 'Stochastic Gradient Descent'. in *Understanding Machine Learning: From Theory to Algorithms*. ed. by Anon Cambridge: Cambridge University Press, 150-166