

## CSC8635: The Titanic Problem – Callum Simpson B603026

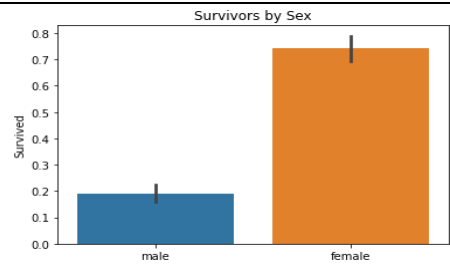
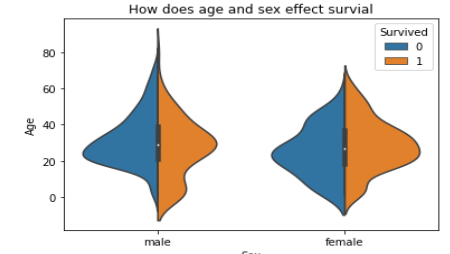
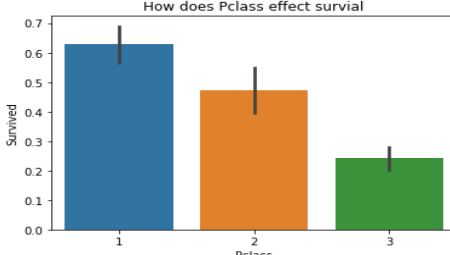
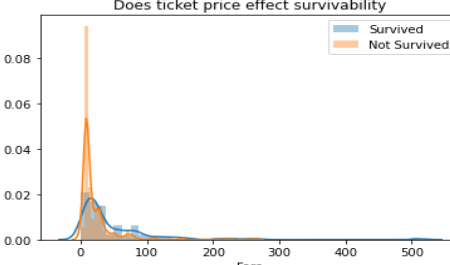
### The problem

On the 15th of April 1912, the RMS Titanic unfortunately collided into an iceberg and sank in the North Atlantic Ocean. Only 722 out of 2,224 people survived the event making it one of the deadliest maritime disasters in modern history. While there is some element of luck to if someone survived it seems some groups were more likely to. The aim of this report will be to use data about those on the Titanic to build a Machine learning model that will be able to accurately predict if a person would have survived the Titanic. [1]

### Data exploring

The data provided for this project has been split into a training set and a test set. Looking at the training data we have 891 samples to work with, each with 12 features. We see that throughout the data we are missing values Age, Cabin, Embarked and Fare. Combined both the training and test database together we see Cabin is missing 78% of its data and Age is missing 21% of its data. We also see we have records for 809 dead and 500 survived.

### Exploration of features

<b>Sex</b> - There was a lot less women aboard the Titanic than men however a lot more females survived the disaster. Over 70% of females survived the Titanic where less than 20% of males survived the Titanic.	
<b>Age</b> - We see that there are certain age groups that have a chance of surviving. This and the sex survival rate suggest that the phrase "Women and children first!" does seem to hold.	
<b>Pclass</b> - The ticket class that the person belonged to could be used to assume what social tier they belong to. We see the higher the ticket class the higher chance of survival. This clearly shows that PClass is a discriminative feature.	
<b>Fare</b> - We see that people who spent less money on a ticket had a lower survival rate when compared to those who paid more.	

**Name** - Name shouldn't have much effect on if a person lives or not.

**Cabin** - As we saw cabin is missing over 78% of its data across both datasets. I will probably drop this value from the dataset due to cleaning it will take too many assumptions.

**SibSp and Parch** - As both relate to if a person had any family aboard the Titanic, I decided that I would use these to work how many people a person was with.

### Cleaning data

As we only had one or two values missing for Embarked and Fare, I decide for Embark I would just use the most common value (so S) and for Fare I would use the median fair value. For Age I decided to use the median age for that person's sex and Pclass (originally it was random age between median and sd but this added to many moving parts making reproducibility a bit more difficult). The medians used in the test data where the ones gotten from the training data.

### Feature engineering

**Family** – As SibSp and Parch tell how many people a person is travelling with I decided that I could combined the two together to make one column called relatives. This led me to create 3 more features Alone (the person is not travelling with anyone), Small Family (If a person has one to 3 relatives on the titanic) and Large Family (If a person had 4 more relatives on the titanic).

**Titles** – Age and sex are big for determining survival rates. Titles can be aged based, so I wanted to work what each person's title was. I abstracted the title information from the name feature to get everybody title (I.e. Mr, Miss). After some cleaning and combining I was able to get 5 distinct name groups.

### Preparing the data

As a lot of algorithms require inputs to be numeric variables, I performed one hot encoding on the following Categorical variable 'Pclass', 'Sex', 'Embarked' and 'Title'. Some classifiers have difficulty working with Categorical variables so by using One hot encoding we can turn Categorical Data which do not have any relationship between categories in a classifier by turning each categorical data variable into a sperate binary variable. I used this over Integer Encoding as allowing the model to assume a natural ordering between categories may result in mediocre performance [2].

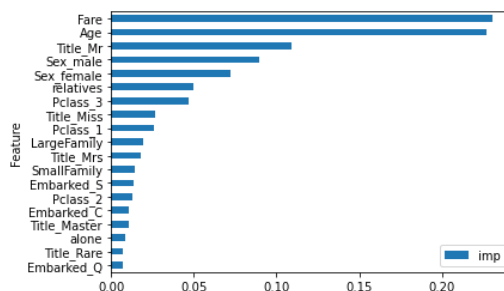
For all the continuous data (age, fair, relatives) I decided to keep my data as continuous data as binned (categorizing) it as it may of lead to the loss of lots of information and poor results.[3]. However, doing this meant I had data with different units, so I applied a standard scaler on them. If I didn't then I would be introducing bias as variables that are measured at different scales do not contribute equally to the analysis [4]. I did not scale the one hot encoded variable as this would have cause issues binary representation of a features. Once done that I dropped the following from my datasets as they were no longer required 'PassengerId', 'Name', 'Cabin', 'Ticket', 'Parch', 'SibSp' and 'relatives'. After doing this I had 18 unique features to work with.

### Feature selection

For this project I have used two sets of features. A negative of One hot encoding is in “bloat” a data set and cause problems due to an increase dimensionality [5]. To combat this, I wanted to use a method to reduce the number of features to allow my ML algorithms to train faster and to help improve accuracy [6]. I picked feature importance as it is widely used technique in the data science community and Recursive Feature Elimination [7].

**The Feature importance method** – Feature importance works by giving us a score for each feature in the data, the higher the score the more relevant that feature is towards the output variable. This approach is good

because it is non-parametric. Sklearn says that Random Decision Trees are best for classification due to its good accuracy, robustness, and ease of use and should give me a subset of features that will reduce the chance of polluting the model [8].



Select- 'Age', 'Fare', 'relatives', 'LargeFamily', 'Pclass\_1', 'Pclass\_3', 'Title\_Miss', 'Title\_Mr', 'Sex\_female', 'Sex\_male'

**Recursive Feature Elimination CV (RFECV)** - A wrapper-type feature selection algorithm works by searching for a subset of features by fitting the machine learning algorithm used in the core of the model, ranking features by importance, discarded the least key features, and then re-fitting the model. This process is repeated until a specified number of features remains. Its good as a feature that is weak at the start may turn out to be one of the best features at the end. [9] [18]. To avoid an issue where to few features are selected, I set a minimum number of features to be selected to be 5 (meaning it could give me more than 5 features back). This run gave me

Age	Fare	relatives	Pclass_3	Title_Mr	Title_Rare	Sex_male
-----	------	-----------	----------	----------	------------	----------

## Classifiers

For each of my classifier I have ran each features selection twice, once with the default hyper parameters and once whilst doing a hyper parameter sweep using GridSearchCV [10]. All my classifiers were built using 5 StratifiedKFold so each fold had the same percentage of samples for each class, this was done due to the imbalanced in survived[11]. All models have been chosen due to their ability to perform well at classifying.

**Logistic regression classifier** - Logistic regression can be used to find the probability of a normally binary outcome by utilizing a logistic (or sigmoid) functions. I will be focusing on Binary Logistic Regression. Hyper parameters where solver (the solver system used), penalty (the penalty system used), c (controls the penalty strength) and max\_iter(Max iterations taken for the solvers to converge). [12]

**Random Forest Classifier** - It works by building an ensemble of decision trees usually trained with the bagging method. Random forests work by splitting the data into random samples, create a tree for each sample, then perform a vote on each prediction made by each tree what evert prediction got the most votes is the final prediction. Hyper parameters used n\_estimators(Number of trees), criterion (used for split),max\_featuers(max number of features per split), min\_samples\_leaf (The minimum number of samples required to be at a leaf node.), max\_depth(The maximum depth of the tree).[13]

**eXtreme Gradient Boosting Classifier** - Based off gradient boosting decision trees algorithms. Ensembles are constructed from decision tree models. Boosting is an ensemble machine learning model in which weak Trees are added to the ensemble one at a time and then fit to correct the prediction errors made by prior models. The reason I am looking at XGBoost over gradient boosting is that it applies a better regularization technique that

helps reduce overfitting. Hyperparameters used are `n_estimators`, `learning_rate` (contribution that each model has on the ensemble prediction) and `subsample` (samples used to fit each tree). [14]

**k-Nearest Neighbor** - Supervised classifier that Classifies each data point by analyzing its nearest neighbors from the training set. The current data point is assigned the class most found among its neighbors. It also works well with a small number of features [15]. Hyperparameters used are `neighbors` (number of neighbors), `p` (power parameter), `algorithm` (used to work out the nearest Neighbour).

## Results

To get my results I first pass X train and Y train data into a model that has been selected with the best hyperparameters (or the default ones). Once I had fit the model using that data I used the X test data to make a prediction. This prediction was compared then compared with the grounded truth (Y test) to work out a confusion matrix. As survival is an imbalanced classifier, I used the confusion matrix to calculate the following to help me perform a better evaluation [16] [17]

**Precision** - True Positives / (True Positives + False Positives). What proportion of possible identification where correct? So The percentage of people that were predicted as surviving that where correct.

**Recall** - True Positive / True Positives + False negatives) The ratio of correctly predicted positive observations to all observations of the class. The proportion of passengers that were correctly identified to have survived.

**Specific** - True Negatives/(True Negatives + False positives). The proportion of negatives, which got predicted as the negative. The proportion of positive passengers that were correctly identified to have perished.

**f1 score** - balance between precision/recall. f1 Score might be a better measure to use as we have an uneven class distribution (Not survived to survived).

Default Features						
	Model	Accuracy	Recall	precision	Specificity	f1
0	def_LogR	0.763158	0.727848	0.672515	0.784615	0.699088
0	def_LogR_hypP	0.770335	0.734177	0.682353	0.792308	0.707317
0	def_RFC	0.748804	0.677215	0.664596	0.792308	0.670846
0	def_RFC_hyp	0.779904	0.708861	0.708861	0.823077	0.708861
0	def_XGB	0.748804	0.658228	0.670968	0.803846	0.664537
0	def_XGB_hyp	0.767943	0.664557	0.704698	0.830769	0.684039
0	def_k_NN	0.765550	0.708861	0.682927	0.800000	0.695652
0	def_k_NN_hyp	0.765550	0.683544	0.692308	0.815385	0.687898
Feature Importance						
	Model	Accuracy	Recall	precision	Specificity	f1
	Imp_LogR	0.782297	0.727848	0.705521	0.815385	0.716511
	Imp_LogR_hypP	0.782297	0.740506	0.700599	0.807692	0.720000
	Imp_RFC	0.741627	0.677215	0.652439	0.780769	0.664596
	Imp_RFC_hyp	0.760766	0.658228	0.693333	0.823077	0.675325
	Imp_XGB	0.744019	0.670886	0.658385	0.788462	0.664577
	Imp_XGB_hyp	0.755981	0.664557	0.681818	0.811538	0.673077
	Imp_k_NN	0.758373	0.677215	0.681529	0.807692	0.679365
	Imp_k_NN_hyp	0.791866	0.715190	0.729032	0.838462	0.722045
Recursive feature elimination with cross-validation						
	Model	Accuracy	Recall	precision	Specificity	f1
0	Rec_LogR	0.784689	0.696203	0.723684	0.838462	0.709677
0	Rec_LogR_hypP	0.775120	0.702532	0.702532	0.819231	0.702532
0	Rec_RFC	0.729665	0.645570	0.641509	0.780769	0.643533
0	Rec_RFC_hyp	0.767943	0.664557	0.704698	0.830769	0.684039
0	Rec_XGB	0.739234	0.658228	0.654088	0.788462	0.656151
0	Rec_XGB_hyp	0.767943	0.658228	0.707483	0.834615	0.681967
0	Rec_k_NN	0.758373	0.658228	0.688742	0.819231	0.673139
0	Rec_k_NN_hyp	0.763158	0.670886	0.692810	0.819231	0.681672

First let's discuss some similarities in results. Accuracy was pretty good throughout all my results; all got an accuracy in the Mid to high 70s. However, that's not we are wanting to look at. We see that all the models have

a slightly better time correctly predicting who died (Spec) than who lived (Recall). As our data isn't 50/50, I will use f1 (mean of precision and recall) to state how well our models perform.

Looking at best model there is one model that consistently got an f1 above 70 and that was logistical regression. Even with no hyper parameters we get some of the best performances out of this classifier. It seems that log's ability to find the probability of a normally binary outcome by utilizing a logistic predictor allowing it better flexibility on how to decide how its going to classify the output. [19].

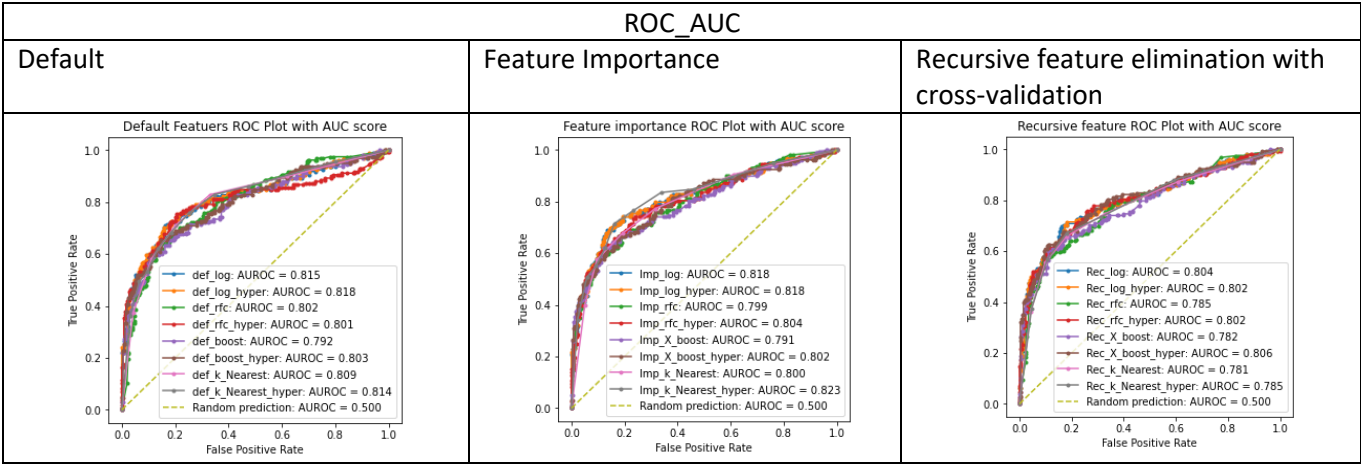
If we were to rank the feature selected by models f1s its clearly Feature importance first as it contains the top 3 f1 models. Default would be second, it seems that using all 19 features that one hot produced has led to a “bloat” in features meaning the models are using redundant data. Surprisingly the worst was Recursive features. In the end it decided to choose 7 different features however it seems like this wasn’t enough. It looks like what it ended up picking lacked information to truly help classify survival.

We see the importance Hyperparameters in these results as in almost all cases a sweep through some hyper parameters has cause an improvement in all sections, if I had more time than is possible that through hyper parameters, we could get some impressive results (however it seems that we have a slight over tuning in).

However, the best model I produce was Feature importance K –NN as it scored high in all categories. So if I had to pick a model it be that, However I had to quickly make a model and didn’t have time for hyper parameters I would pick Feature importance logistical regression due to the fact it good a good model with very little effort.

**Receiver Operator Characteristic (ROC) and Area under Curve (AUC)**

ROC plots True positive rates (Sensitivity) against False positive rates and measure the degree of separability a model can make. AUC is the area under the ROC curve. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0s as 0s and 1s as 1s. [17]. Below is all the ROC and AUC for each of the models that I created. We see that all classifiers created are good at predicting 0s as 0s and 1s as 1s as all have achieved a AUCROC of around 80. The best was In importance feature K-NN with hyper parameters.



## Discussion

We have run several models, and all have returned different results. Looking at the features each selection made we see the following

Imp	Age	Fare	relatives	LargeFamily	Pclass1	Pclass3	Miss	MR	Female	Male	
RFE	Age	Fare	relatives			Pclass3		MR		Male	Rare

We see some overlap, both decided that if a person was male, had relatives onboard, where in P Class of 3, how much they paid, and a MR play an important in working out if a person lived or not. We see that IMP also decide that being a large family, female and being in Pclass 1 play a large part.

If I were to ask the question if I would survive the titanic I would not as I'm a 22-year-old male, a MR, from a large family and I would be in Pclass 3 meaning that I neatly fall into the features that have been deemed the most important to deciding if someone would have not survived. If I were 8, from first class and from a small family then my chances of living go up a lot more so I may have survived the disaster.

Even though one scored better than the other it, both show that there are clear groups of people that were more likely to survive.

However, it should be said that none of my models scored 100% suggesting that whilst some groups did have a higher chance of surviving than others, we must assume that there was some element of luck involved/other factors at play. The Survival of the titanic could be seen in two stages. If you made it onto a lifeboat you were safe. If phrase "woman and children first" is true, then that is why they probably had a high survival rate. However, it is probable that some people who fit these criteria did not make it onto a boat and it is also possible that some older males would have made it onto a lifeboat meaning we would have records that give us the opposite of what we expect. The 2nd survival stage would be then people who did not make it onto a lifeboat but were able to survive long enough for a rescue craft to save them. Whilst making it onto a lifeboat seems to depend on the Sex and Age of the person surviving a sinking ship may require more factors to determine if that person survived that is not in the database. It is also possible that it may just been based on luck.

## Conclusion

In Conclusion, Yes, some groups of people on the titanic where more likely to survive. We have seen this through our EDA, features selections and modellings. The 3 mains groups where children, Females and people form P Class 1. I believe that you could create a model using a good set of features and well optimized hyper parameters and create a something that could probably get a low 80s accuracy at best. I feel like getting higher than that would be near impossible. Whilst its easy now to look over the data and see that certain groups of people had a higher change of survival you must Imagin the amount of fear and panic of the people during the sinking. People would have done what they could to protect them self's and help those around them, meaning that there may be additional factors in deciding if a person did or did not survive the Titanic that we simply will not know. Some of which might just come down to luck.

Please feel free to contact me if you want my code, apologies if I have gone over a word limit. There was no clear number on the spec

## Bibliography

1	Kaggle.com. 2021. <i>Titanic - Machine Learning from Disaster   Kaggle</i> . [online] Available at: < <a href="https://www.kaggle.com/c/titanic">https://www.kaggle.com/c/titanic</a> > [Accessed 6 December 2020].
2	Brownlee, J., 2021. <i>Why One-Hot Encode Data in Machine Learning?</i> . [online] Machine Learning Mastery. Available at: < <a href="https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/">https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/</a> > [Accessed 7 January 2021].
3	O. Naggara, J. Raymond, F. Guilbert, D. Roy, A. Weill and D.G. Altman ., 2011 Analysis by Categorizing or Dichotomizing Continuous Variables Is Inadvisable: An Example from the Natural History of Unruptured Aneurysms / American Journal of Neuroradiology March 2011, 32 (3) 437-440; DOI: [online] Available at: < <a href="https://doi.org/10.3174/ajnr.A2425">https://doi.org/10.3174/ajnr.A2425</a> >
4	Team, T. and Team, T., 2021. <i>How, When, and Why Should You Normalize / Standardize / Rescale Your Data?</i> . [online] Towards AI — The Best of Tech, Science, and Engineering. Available at: < <a href="https://towardsai.net/p/data-science/how-when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff">https://towardsai.net/p/data-science/how-when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff</a> > [Accessed 14 January 2021].
5	DictVectorizer?, W., Joshi, P. and singh, k., 2021. <i>When to use One Hot Encoding vs LabelEncoder vs DictVectorizer?</i> . [online] Data Science Stack Exchange. Available at: < <a href="https://datascience.stackexchange.com/questions/9443/when-to-use-one-hot-encoding-vs-labelencoder-vs-dictvectorizer#:~:text=One%2DHot%2DEncoding%20has%20the,with%20the%20curse%20of%20dimensionality.&gt;">https://datascience.stackexchange.com/questions/9443/when-to-use-one-hot-encoding-vs-labelencoder-vs-dictvectorizer#:~:text=One%2DHot%2DEncoding%20has%20the,with%20the%20curse%20of%20dimensionality.&gt;</a> > [Accessed 14 January 2021].
6	Introduction to Feature Selection methods with an example (or how to select the right variables?) , I. and Kaushik, S., 2021. <i>Feature Selection Methods   Machine Learning</i> . [online] Analytics Vidhya. Available at: < <a href="https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/#:~:text=Top%20reasons%20to%20use%20feature,the%20right%20subset%20is%20chosen.&gt;">https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/#:~:text=Top%20reasons%20to%20use%20feature,the%20right%20subset%20is%20chosen.&gt;</a> > [Accessed 14 January 2021].
7	Packt Hub. 2018. <i>4 ways to implement feature selection in Python for machine learning   Packt Hub</i> . [online] Available at: < <a href="https://hub.packtpub.com/4-ways-implement-feature-selection-python-machine-learning/">https://hub.packtpub.com/4-ways-implement-feature-selection-python-machine-learning/</a> > [Accessed 16 January 2021].
8	Scikit-learn.org. 2021. <i>1.13. Feature selection — scikit-learn 0.24.1 documentation</i> . [online] Available at: < <a href="https://scikit-learn.org/stable/modules/feature_selection.html#univariate-feature-selection">https://scikit-learn.org/stable/modules/feature_selection.html#univariate-feature-selection</a> > [Accessed 16 January 2021].
9	Brownlee, J., 2020. <i>Recursive Feature Elimination (RFE) for Feature Selection in Python</i> . [online] Machine Learning Mastery. Available at: < <a href="https://machinelearningmastery.com/rfe-feature-selection-in-python/#:~:text=Recursive%20Feature%20Elimination%2C%20or%20RFE%20for%20short%2C%20is%20a%20feature,columns%2C%20like%20an%20excel%20spreadsheet.&amp;text=Feature%20selection%20refers%20to%20techniques,(columns)%20for%20a%20dataset.&gt;">https://machinelearningmastery.com/rfe-feature-selection-in-python/#:~:text=Recursive%20Feature%20Elimination%2C%20or%20RFE%20for%20short%2C%20is%20a%20feature,columns%2C%20like%20an%20excel%20spreadsheet.&amp;text=Feature%20selection%20refers%20to%20techniques,(columns)%20for%20a%20dataset.&gt;</a> > [Accessed 16 January 2021].
10	Scikit-learn.org. 2021. <i>sklearn.model_selection.GridSearchCV — scikit-learn 0.24.1 documentation</i> . [online] Available at: < <a href="https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html">https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html</a> > [Accessed 15 January 2021].
11	Scikit-learn.org. 2021. <i>sklearn.model_selection.StratifiedKFold — scikit-learn 0.24.1 documentation</i> . [online] Available at: < <a href="https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html">https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html</a> > [Accessed 16 January 2021].
12	Scikit-learn.org. 2021. <i>sklearn.linear_model.LogisticRegression — scikit-learn 0.24.1 documentation</i> . [online] Available at: < <a href="https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html">https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html</a> > [Accessed 13 January 2021].
13	Scikit-learn.org. 2021. <i>sklearn.ensemble.RandomForestClassifier — scikit-learn 0.24.1 documentation</i> . [online] Available at: < <a href="https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html">https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html</a> > [Accessed 15 January 2021].
14	Brownlee, J., 2020. <i>Extreme Gradient Boosting (XGBoost) Ensemble in Python</i> . [online] Machine Learning Mastery. Available at: < <a href="https://machinelearningmastery.com/extreme-gradient-boosting-ensemble-in-python/#:~:text=Number%20of%20Features-">https://machinelearningmastery.com/extreme-gradient-boosting-ensemble-in-python/#:~:text=Number%20of%20Features-</a>

	,Extreme%20Gradient%20Boosting%20Algorithm,or%20regression%20predictive%20modeling%20problems.&ext=Trees%20are%20added%20one%20at,errors%20made%20by%20prior%20models.> [Accessed 19 January 2021].
15	Scikit-learn.org. 2021. <i>1.6. Nearest Neighbors — scikit-learn 0.24.1 documentation</i> . [online] Available at: < <a href="https://scikit-learn.org/stable/modules/neighbors.html">https://scikit-learn.org/stable/modules/neighbors.html</a> > [Accessed 19 January 2021].
16	Medium. 2018. <i>Accuracy, Precision, Recall or F1?</i> . [online] Available at: < <a href="https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9">https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9</a> > [Accessed 16 January 2021].
17	Narkedede, S., 2021. <i>Understanding AUC - ROC Curve</i> . [online] Medium. Available at: < <a href="https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5">https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5</a> > [Accessed 26 June 2018].
18	Scikit-learn.org. 2021. <i>sklearn.feature_selection.RFECV — scikit-learn 0.24.1 documentation</i> . [online] Available at: < <a href="https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html#sklearn.feature_selection.RFECV">https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html#sklearn.feature_selection.RFECV</a> > [Accessed 19 January 2021].
19	Medium. 2021. <i>Random Forest vs Logistic Regression</i> . [online] Available at: < <a href="https://medium.com/@bemali_61284/random-forest-vs-logistic-regression-16c0c8e2484c">https://medium.com/@bemali_61284/random-forest-vs-logistic-regression-16c0c8e2484c</a> > [Accessed 21 January 2021].