University of Massachusetts Amherst

Manning College of Information & Computer Sciences

COMPSCI 589: MACHINE LEARNING FALL 2023

Titanic Survival Prediction Model

Yash Kamoji
34032599
ykamoji@umass.edu

1 INTRODUCTION

The RMS Titanic was a British passenger liner that sank in the North Atlantic Ocean in the early morning hours of 15 April 1912, after it collided with an iceberg during its maiden voyage from Southampton to New York City. There were an estimated 2,224 passengers and crew aboard the ship, and more than 1,500 died, making it one of the deadliest commercial peacetime maritime disasters in modern history. The RMS Titanic was the largest ship afloat at the time it entered service and was the second of three Olympic-class ocean liners operated by the White Star Line. The Titanic was built by the Harland and Wolff shipyard in Belfast. Thomas Andrews, her architect, died in the disaster.

1.1 PROBLEM

We aim to address the task of forecasting the survival outcomes of passengers aboard the Titanic. The main challenges are to leverage an ensemble of machine learning algorithms to predict whether a given passenger survived or did not survive the sinking of the ship. An essential aspect of the problem is the data preparation for the models to give highly accurate results, especially, figuring out which features to pick and uncover intricate patterns as needed. Finally, the last hurdle involves finding the best set of models with their hyper parameters, and model performance evaluations.

1.2 DATASET

The dataset can be found here `https://www.kaggle.com/c/titanic/data`. The training set consists of **891** passengers with the ground truth indicator whether they survived or not. The test set contains **328** passengers whom we need to identify if they survived or not.

1. Survival : 0 or 1. Indicates whether the passenger survided or not.
2. Pclass : 1, 2 or 3. Ticket class.
3. Sex
4. Age
5. Sibsp : # of siblings / spouses aboard the Titanic.
6. Parch : # of parents / children aboard the Titanic.
7. Ticket : Ticket number.
8. Fare : Passenger fare.
9. Cabin : Cabin number.
10. Embarked : C = Cherbourg, Q = Queenstown, S = Southampton. Port of Embarkation.

1.3 OBJECTIVES

In this report, we aim to predict the survivors of the Titanic by exploring, cleaning, and engineering features, and finally developing model(s) capable of predicting survivors with high accuracy. The objectives can be broken down into below steps:

1. Data Prepossessing
   - 1.1 Cleaning
   - 1.2 Transformation
2. Model implementation
   - 2.1 Kernels, Forest & NN models
   - 2.2 Hyperparamter tuning
3. Model Performance
   - 3.1 Accuracy
   - 3.2 Performance Comparison

2 METHODOLOGY

2.1 DATA EXPLORATION

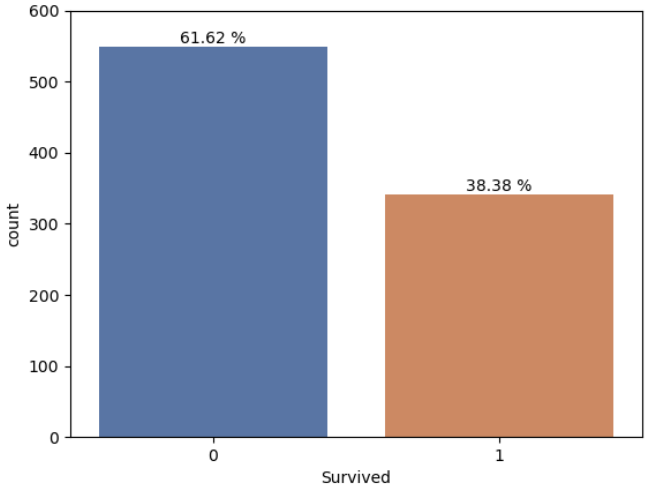
We first take a quick glance at the data (features).

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

	Na	%
Pclass	0	0.00
Name	0	0.00
Sex	0	0.00
Age	263	20.09
SibSp	0	0.00
Parch	0	0.00
Ticket	0	0.00
Fare	1	0.08
Cabin	1014	77.46
Embarked	2	0.15

Features with missing data:

1. **Age** (20%): A moderate amount of data is missing, however, it would be useful to fill the missing ages.
2. **Cabin** (77%): Too many missing data. We just drop it since filling will introduce a significant amount of noise.
3. **Fare** (1 data point): Since only one value is missing we can fill it with mean.
4. **Embarked** (2 data points): Since only two values are missing we can fill them with the majority value S.



We can see that 38% out of the training-set survived the Titanic. So our predictions on the testing dataset should also follow similar trend and have less survivors.

2.2 NAME

While *Name* in itself cannot be used as a feature, we can still get use full information out of it by pulling out the titles from names.

Most passengers have one of four titles:

1. Master
2. Miss
3. Mr
4. Mrs

We can join *Mlle* with *Miss* since they both typically refer to an unmarried female.

Mme with Mrs since they both typically refer to a married female.

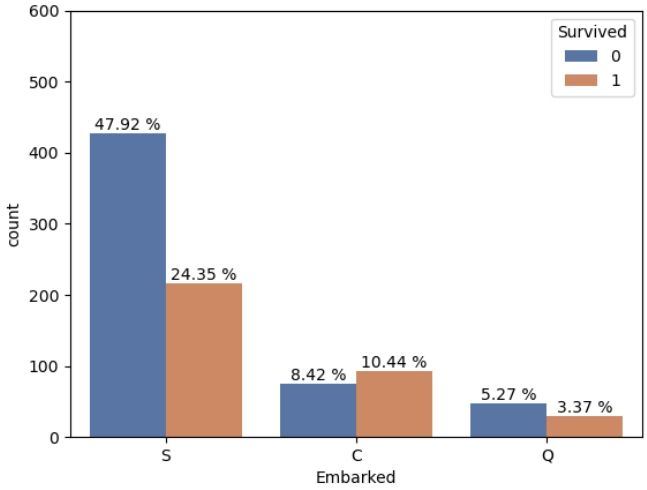
Between *Ms* and *Miss*, *Ms* is a more generic title that can either refer to *Miss* or *Mrs* but given that *Miss* is more frequent we'll go with that.

For the remaining titles, we are going to gather them all in a title named *Rare* to convey that these passengers have a unique social status.

Title	Name
Mr	757
Miss	260
Mrs	197
Master	61
Rev	8
Dr	8
Col	4
Ms	2
Major	2
Mlle	2
Sir	1
Capt	1
Mme	1
Lady	1
Jonkheer	1
Dona	1
Don	1
the Countess	1

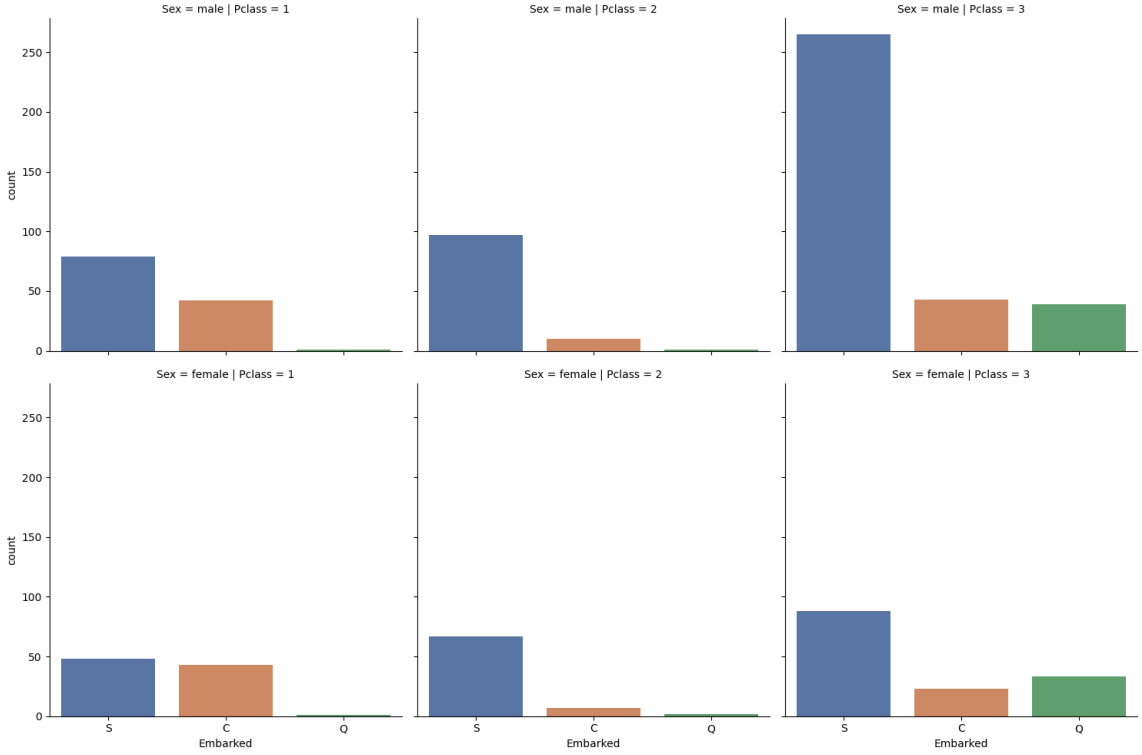
2.3 EMBARKED

It would be interesting to check was a useful feature. We can see that passengers from different ports have varying chances of survival.



From the above chart we definitely see some correlation between survival and *Embarked*. On average, passengers joining from *S* port have higher chances of being both victim and survivors. Passengers joining from *Q* port have lower chances of being both victim and survivors.

However, intuitively it doesn't really make sense since port shouldn't make any difference on whether someone survives or not.

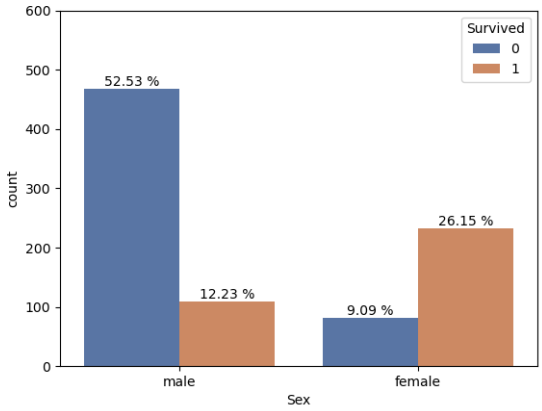


			Total	Survived	Rate
Embarked	Pclass	Sex			
C	1	female	43	42	0.976744
		male	42	17	0.404762
	2	female	7	7	1.000000
		male	10	2	0.200000
	3	female	23	15	0.652174
		male	43	10	0.232558
Q	1	female	1	1	1.000000
		male	1	0	0.000000
	2	female	2	2	1.000000
		male	1	0	0.000000
	3	female	33	24	0.727273
		male	39	3	0.076923
S	1	female	48	46	0.958333
		male	79	28	0.354430
	2	female	67	61	0.910448
		male	97	15	0.154639
	3	female	88	33	0.375000
		male	265	34	0.128302

We see that in *S*, a high number of 3^{rd} class males embarked on the Titanic, so a low survival rate is to be expected. A high number of 1^{st} class passengers embarked in *C*, so we see a high survival rate.

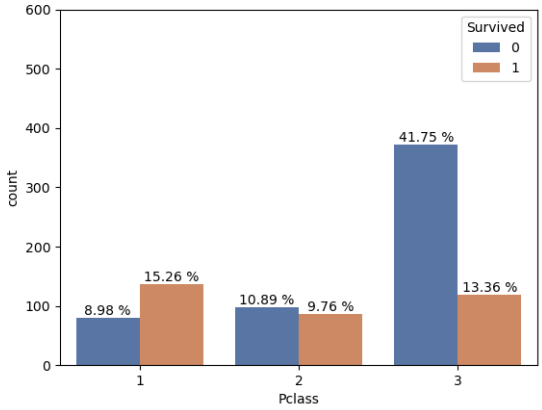
In *Q*, the vast majority of passengers are 3^{rd} class passengers. The number of males and females is comparable. Given these conditions, the survival rate should be correlated only to *Sex*, and that is true for females. Males however end-up with half the survival rate. Though this is a sample of 42 males in a population of 577, it's probably not correlated enough to survival, so we can **disregard** this from the feature selections list. We will though, use it this for *Age* prediction.

2.4 Sex



Females have much higher survival rates as expected, all we have to do is one-hot encode *Sex* so all machine learning algorithms can handle it.

2.5 Pclass



Passengers with higher ticket class (lower *Pclass*) have higher chances of survival:

1st class: 15.26%

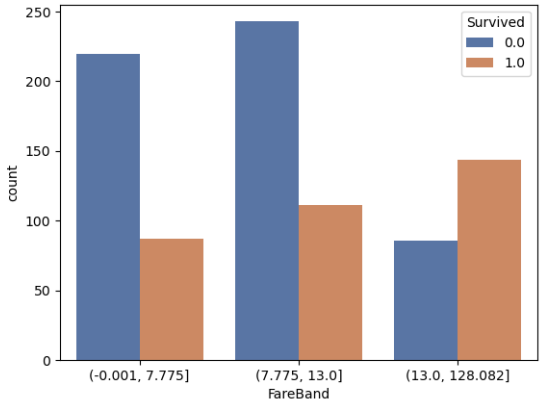
2nd class: 9.76%

3rd class: 13.36%

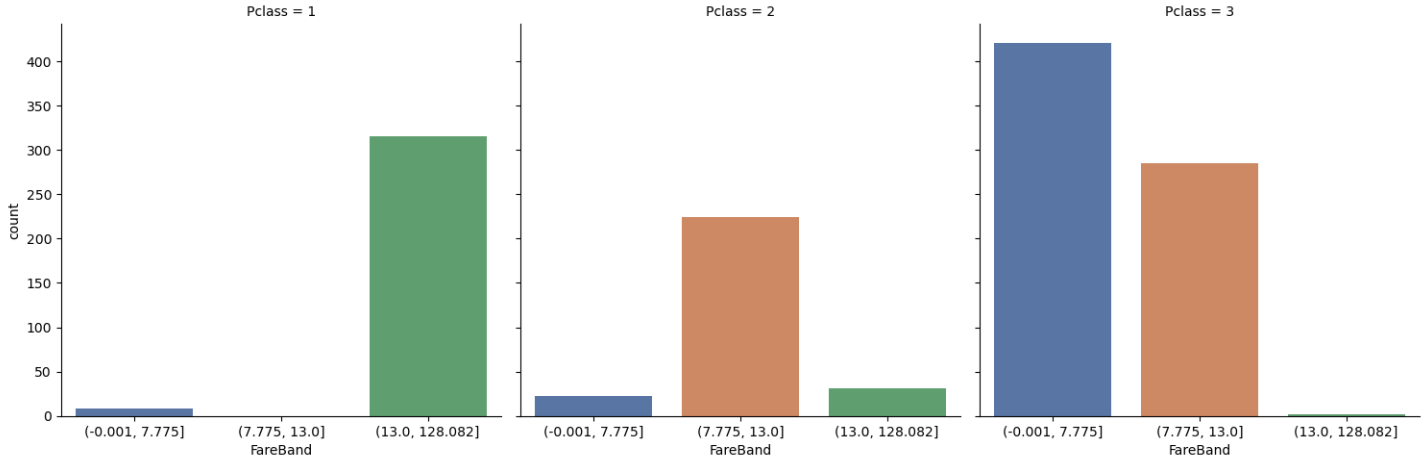
Keep the feature as-is, *Pclass* is already encoded in integers (one-hot encoding would leave out the natural order of ticket class which is an important element of the feature).

2.6 FARE

We will group *Fare* into bands to analyse this feature.



We see that as Fare increases, the survival rate also increases. However, it's very likely that this feature is highly related to *Pclass*, lower *Pclass* should translate to higher *PassengerFare*.



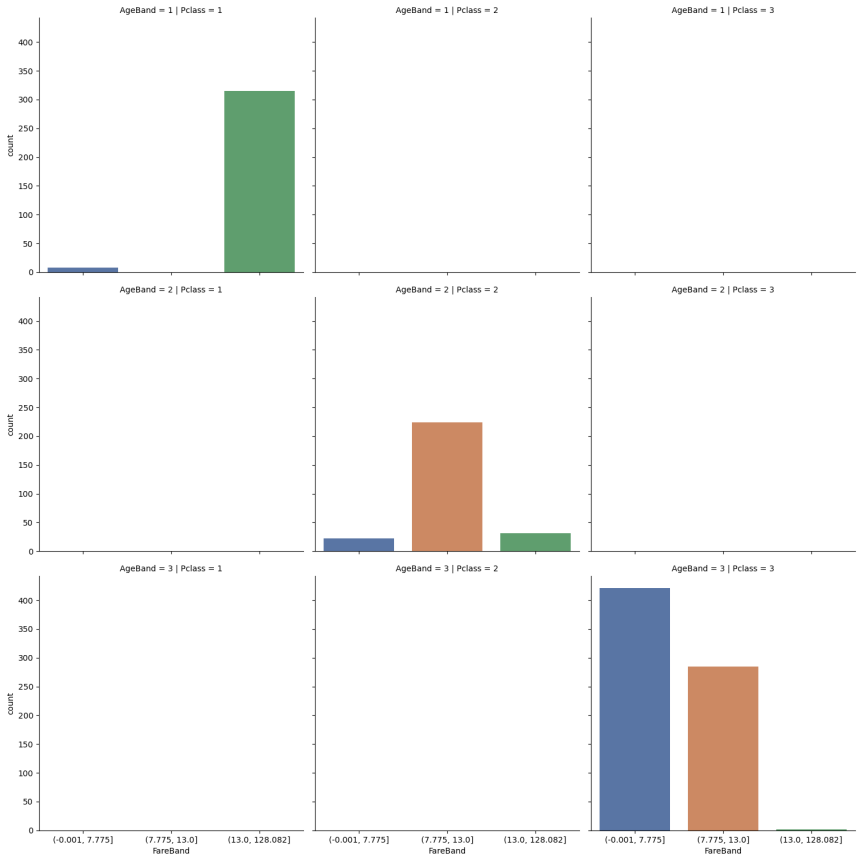
FareBand describes *Pclass* 1 and 2 almost perfectly. However, there so many passengers with *Pclass* = 3 tickets with *fares* expected for *Pclass* = 2. Let's compare the mean passenger fare and survival for passengers in 2^{nd} and 3^{rd} class in the same fare range ((7.775, 13.0]).

		PassengerFare
		mean
FareBand	Pclass	
(7.775, 13.0]	2	11.648066
	3	8.225494

Despite belonging to the same fare band, 3^{rd} class passengers paid on average less than 2^{nd} class passengers. There is a clear boundary between the two, just not enough to segment them into different bands. We can also check if there is any correlation between 3^{rd} class passengers that paid higher fares and survival.

		Survived
		mean
FareBand	Pclass	
(-0.001, 7.775]	3	0.280142
(7.775, 13.0]	3	0.193237

Passengers that overpaid for their tickets shouldn't have substantially different chances of survival. Surprisingly, those that overpaid have lower chances of survival. If we look at the other features and their relationship with these two groups we find something interesting about the distribution of children.



Here the 83 children (AgeBand = 0) with 3^{rd} class ticket in FareBand = (-0.001, 7.775] and only 2 in FareBand = (7.775, 13.0]. Given that children have 60% survival rate, the lack of them in FareBand = (7.775, 13.0] compared to FareBand = (-0.001, 7.775] explains the lower survival rate.

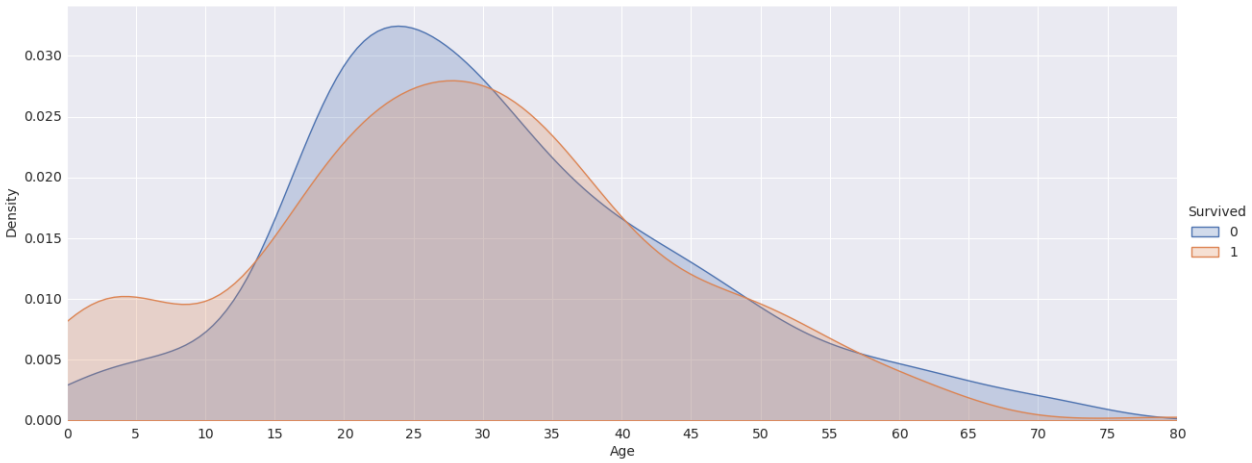
We can also explain why some 3^{rd} class passengers seem to have overpaid for their tickets. However, they in reality didn't. The passengers in FareBand = (-0.001, 7.775] paid less because children had a discounted price lowering the passenger fare and isolating them in FareBand = (-0.001, 7.775].

In conclusion, *Fare* is correlated with the other features, therefore, we'll drop it from the features list.

2.7 Age

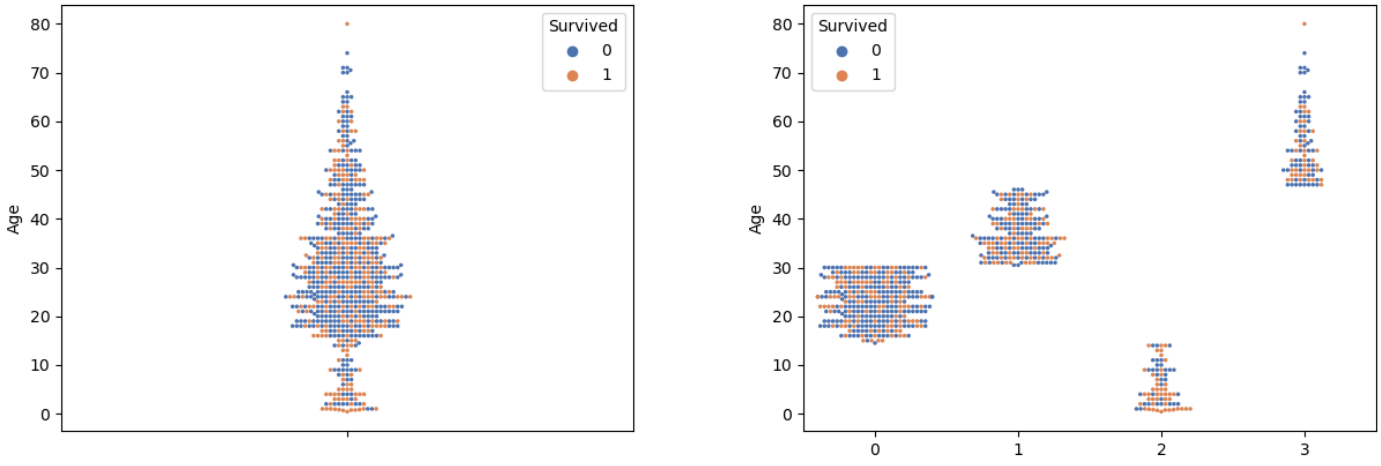
To fill the missing value of Age, we need analyse the hidden pattern with the available age. Since the age data provided in the dataset is possibly inaccurate, instead of predicting exact age, we can instead use a new feature *Age Group* that will be treated a category and hence give better prediction without adding too much distortion to the model.

Observations:



1. Children (age < 5) have much higher chances of survival than any other age group
2. Passengers between the ages of 13 to 30 are more likely to die than to survive
3. From 30 to 60 the survival rate is close to 50%
4. Passengers older than 60 are more likely to die than to survive

Age gives us some important information, particularly for children and younger adults where there is clearly a trend.

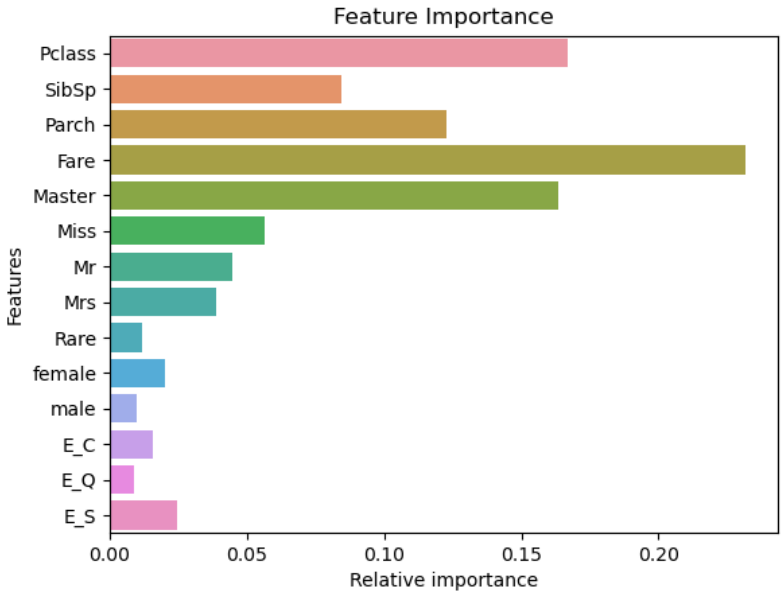


Keeping the previous observations, we'll split *Age* into 4 clusters based on the density graph. The Age groups are approximately,

[0 14 28 44 80]

With these age groups, we create the new feature *Age Group* to both the train and test dataset. Next, we run a classification model to predict the *Age Group* for the missing passengers.

Score	Model
0.579396	RandomForestClassifier
0.570788	DecisionTreeClassifier
0.563123	LogisticRegression
0.541099	KNeighborsClassifier



RandomForestClassifer gives the best average cross validation score and we use this to predict the rest of the missing age groups. Also, it's worth noting that during the prediction, the highest importance of *Age Group* came from *Fare*, *Pclass* & *Master* title. This implies that our age prediction is highly correlated with *Fare* the most.

2.8 Engineering Groups of Passengers

In recent studies, we have uncovered new possibilities of family relationships. Hence, we explore that using SibSp and Parch to engineer family related features.

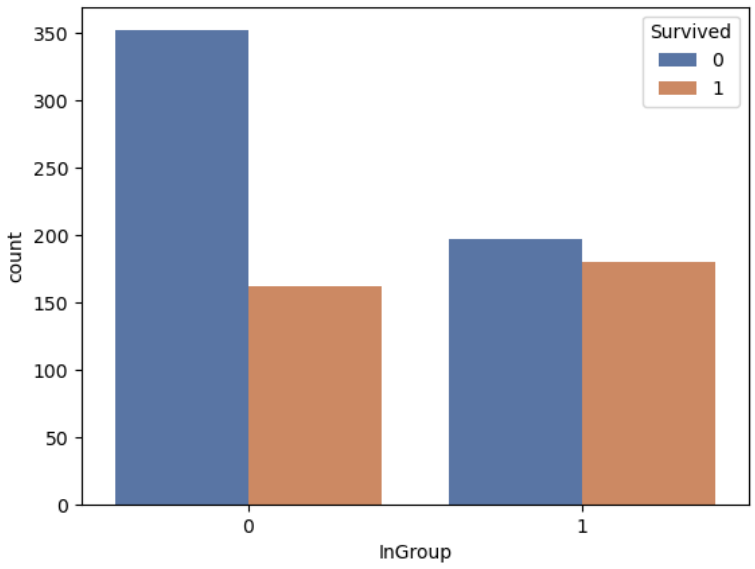
There have been problems using this since *SibSp* and *Parch* contain inconsistencies and that looking at groups of passengers travelling together instead of just families yields better results. With this knowledge we are going to engineer four new features derived from groups of passengers:

- InGroup: the passenger is traveling with other passengers or not.
- InWcg: the passenger is a member of a woman-child group or not
- WcgAllSurvived: whether the members of the woman-child group all survived
- WcgAllDied: whether the members of the woman-child group all died.

We are going to define a group as a set of passengers that all share at least one of the combinations of surname, *PClass*, *ticket number* (excluding the last digit), and *Embarked* or just *ticket number*.

The first condition will identify mostly families while the second will identify groups of friends that share the same ticket, including passengers travelling with families that are not part of the family.

InGroup is 1 if the passenger is in a group (GroupId is not unique); otherwise is 0.




```
InGroup
0    0.315175
1    0.477454
```

Woman-child-groups are groups of passengers traveling together whose members are either females or boys (males with a Master title). *InWcg* is 1 if the passenger is in a woman-child-group; otherwise is 0.

For a given passenger in a woman-child-group, *WcgAllSurvived* is equal to 1 if all members of that group survived; otherwise is 0. Note that passengers from the test set are ignored. *WcgAllSurvived* is based on the training set data only.

WcgAllDied is just the opposite of *WcgAllSurvived*. For a given passenger in a woman-child-group, *WcgAllDied* is equal to 1 if all members of that group died; otherwise is 0.

2.9 Conclusion

We have explored all the features and finalized the ones which are important for predicting survivors of the test dataset passengers. The features we are going to use are,

- Age Group
- Pclass
- Sex
- InGroup
- InWcg
- WcgAllSurvived
- WcgAllDied

3 Implementation Approach

In all the approaches, normalizing the features to converge faster. Since we have encoded all the features in binary or ordinal categories, normalization will make use use less epochs to get maximum accuracy.

3.1 Fixed Shape universal approximators

3.1.1 Model Initialization

To get a baseline of the model, we first initialize the SVM model with a linear kernel.

3.1.2 Hyperparameter Tuning

Regularization Parameter : Tune the regularization parameter C using cross-validation. A smaller C allows for a wider margin, but too small may lead to underfitting, while a large C might result in overfitting.

Degree: Controls the degree of the polynomial used in the kernel function. It determines the complexity of the decision boundary. Since we have normalized the features, degree will be efficiently optimized.

Gamma : The Kernel coefficient. Defines how far the influence of a single training example reaches. Low values mean a broader influence, while high values mean a narrower influence. The choice of gamma depends on the specific dataset and the desired trade-off between model complexity and generalization. It is a critical parameter for non-linear kernels.

3.1.3 Feature Engineering

We use the in-built Kernel Trick so that we don't have to explicitly feature engineering. However, domain-specific feature selection or transformation can be applied.

3.1.4 Cross-Validation

We use k-fold cross-validation to assess the model's performance on different subsets of the data. This helps ensure the model's generalization.

3.2 Neural network based universal approximators

3.2.1 Model Initialization

Initialize a neural network with an appropriate architecture (number of layers, neurons per layer, activation functions like Relu, Tanh).

3.2.2 Hyperparameter Tuning

Learning Rate: Tune the learning rate to control the size of the steps taken during optimization. Too high may cause overshooting, and too low may slow down convergence.

Batch Size: Determine the number of samples used in each iteration. Smaller batches provide a form of regularization and may speed up convergence. We have less than 1000 data points, hence batch size is less important here and so, we don't optimize this.

Number of Hidden Layers and Neurons: Experiment with different architectures to find the optimal trade-off between model complexity and generalization. The number of layer will impact the decisions / propogation for all the categorical features of our dataset.

3.2.3 Feature Engineering

Activation Functions: We should choose appropriate activation functions for hidden layers based on the nature of the features.

3.2.4 Cross-Validation

We employ k-fold cross-validation to evaluate the model's performance. This will be crucial for understanding how well the model generalizes to new data.

3.3 Tree-based approaches

3.3.1 Model Initialization

Random Forest: Initialize a Random Forest model with a suitable number of trees.

3.3.2 Hyperparameter Tuning

Number of Trees (n_estimators): Tune the number of trees in the forest. A higher number may reduce overfitting but increases computation time. Since we have less features, we can tune with a larger range.

Max Depth of Trees (max_depth): Control the maximum depth of each tree. Deeper trees can capture more complex relationships but may lead to overfitting. Since we have less features, we can tune with a larger range.

Max_features: The number of features to consider when looking for the best split. The choice of max_features impacts the diversity among the trees. Since we did transformation on all features, we want to make sure almost all the time the decision trees will use all features.

3.3.3 Feature Engineering

Encoding categorical variables appropriately using one-hot encoding.

3.3.4 Cross-Validation

Employed k-fold cross-validation to evaluate the trees performance on different subsets of the data, providing insights into its generalization capability.

4 Result

4.1 Fixed Shape universal approximators

Using the *logistic Regression* (only for comparison) & *SVC* model for this approach. Kernal method used here is *'rbf'*(gave better validation accuracy than others.)

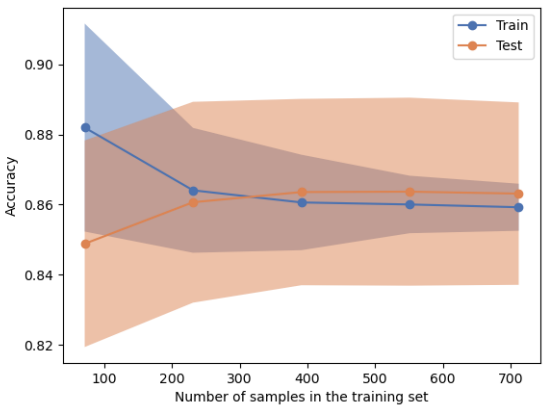
Model Name	Accuracy	Std
SVC	0.854097	0.008399
LogReg	0.820426	0.004199

Turning two parameters: *gamma* & *degree* using GridSearchCV.
For C, manually tuning and finding the optimum value as 0.1.

```
Model SVC

Fitting 5 folds for each of 6 candidates, totalling 30 fits
Model with rank: 1
Mean validation score: 0.8429 (std: 0.0163)
Parameters: {'gamma': 1, 'degree': 2}
Model with rank: 1
Mean validation score: 0.8429 (std: 0.0163)
Parameters: {'gamma': 1, 'degree': 3}
Model with rank: 3
Mean validation score: 0.8384 (std: 0.0147)
Parameters: {'gamma': 2, 'degree': 2}
Model with rank: 3
Mean validation score: 0.8384 (std: 0.0147)
Parameters: {'gamma': 2, 'degree': 3}
```

The best model has a mean cross validation score of 84.3 %.
Evaluating the best model performance using the *Accuracy* graph.



4.2 Neural network based universal approximators

Employing two different models for this approach. The first is modeled using Pytorch and creating an simple architecture from ground up with *Relu* as hidden layers and a *sigmoid* activation. Using *KFold* validation for this model to find the best hyper parameters : *'n_hidden'*, 'd_hidden', *'lr'* & *'optim'*.

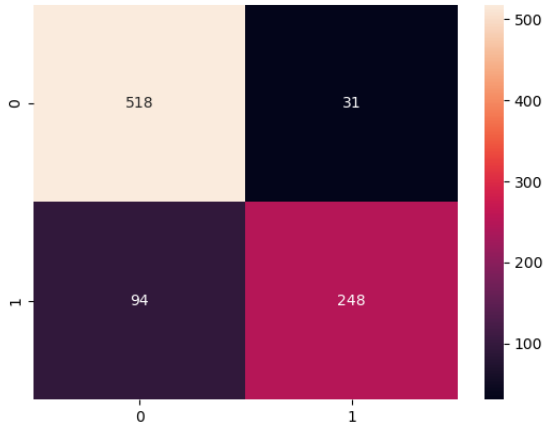
```
Best model
Parameters: {'n_hidden': 10, 'd_hidden': 7, 'lr': 0.01, 'optim': 'Adam'}
Mean CV accuracy: 0.8552
```

In the second model, using a Multi-layer Perceptron classifier and find the optimum hyperparameters: *'activiation'* & *'hidden_layer_size'* using GridSearchCV.

Model Name	Accuracy	Std
MLP	0.841774	0.019375

```
Fitting 5 folds for each of 30 candidates, totalling 150 fits
Model with rank: 1
Mean validation score: 0.8518 (std: 0.0194)
Parameters: {'activation': 'relu', 'hidden_layer_sizes': 6}
Model with rank: 2
Mean validation score: 0.8507 (std: 0.0221)
Parameters: {'activation': 'relu', 'hidden_layer_sizes': 14}
Model with rank: 3
Mean validation score: 0.8507 (std: 0.0271)
Parameters: {'activation': 'relu', 'hidden_layer_sizes': 13}
```

Evaluating the best model performance using a *confusion matrix*.



We see an balanced accuracy of 86.7 %.

4.3 TREE-BASED APPROACHES

Using an ensemble of tree based models for this approach. *RandomForest, XGBoost, DecisionTree, ExtraTrees, GradientBoost, AdaBoost.*

First finding the best model with the default parameters. Using the K-fold validation, the *RandomForest* gives the best average validation score and hence we pick this to tune the hyperparameters.

Model Name	Accuracy	Std
RandomForest	0.846240	0.009655
XGBoost	0.842873	0.013837
DecisionTree	0.841751	0.010997
ExtraTrees	0.841751	0.012598
GradientBoost	0.839506	0.015141
AdaBoost	0.801347	0.004762

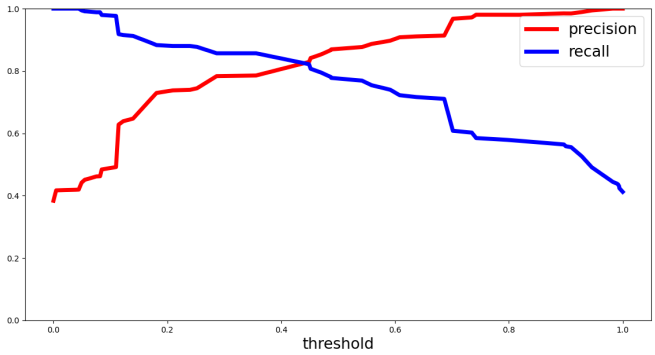
Now we search for the optimum hyperparameters: *'max_depth'* & *'max_features'* using GridSearchCV.

```
Model RandomForest

Fitting 5 folds for each of 60 candidates, totalling 300 fits
Model with rank: 1
Mean validation score: 0.8485 (std: 0.0167)
Parameters: {'max_depth': 4, 'max_features': 5}
Model with rank: 2
Mean validation score: 0.8474 (std: 0.0204)
Parameters: {'max_depth': 4, 'max_features': 7}
Model with rank: 2
Mean validation score: 0.8474 (std: 0.0204)
Parameters: {'max_depth': 4, 'max_features': 8}
Model with rank: 2
Mean validation score: 0.8474 (std: 0.0204)
Parameters: {'max_depth': 4, 'max_features': 9}
```

The best model has a mean cross validation score of 84.8 %.

Evaluating the best model performance using the *LearningCurve* graph.

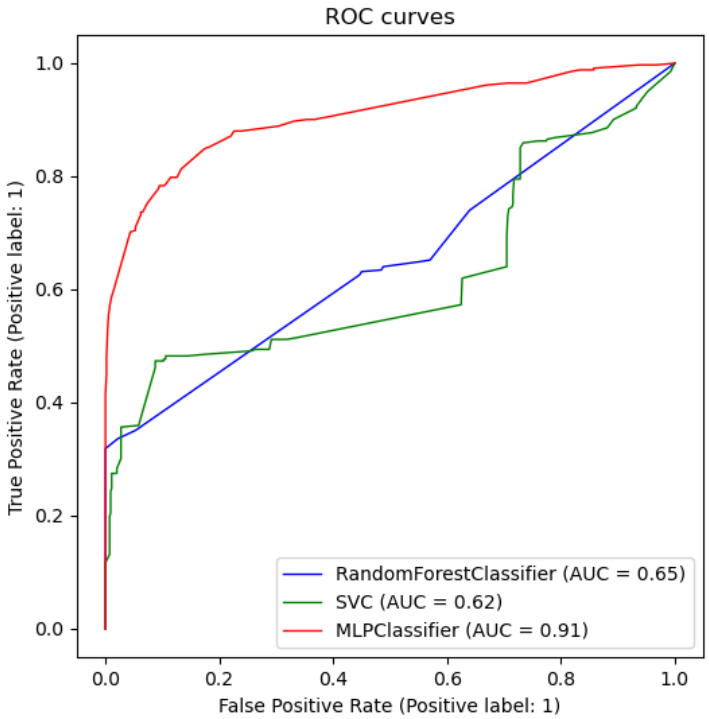


We can see that the RandomForest model's precision increases steadily increases as expected, meanwhile the recall goes down. Controlling this trade off is the next challenge.

4.4 Model performance comparison

Finally, we compare the performance all the best models of the 3 approaches using the *ROC curve* graph.

ROC graph a graphical representation that illustrates the diagnostic ability of a binary classification model across different thresholds for making predictions. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings.



We can see that MLPClassifier reaches the high accuracy quickly than others. On the other hand, RandomForest takes almost a linear & slower approach. SVC has a little bit of variability as it reaches optimum accuracy.

5 CONCLUSION

Model	Average CV Accuaracy	Test Accuracy (Kaggle)
Random Forest	84.6 %	80.382 %
SVC	85.4 %	79.904%
NN	85.41%	79.665%
MLP	85.18%	81.11%

All the models have nearly same accuracy. By doing extensive feature engineering, we see the models are predicts more accurately. This bolsters our understanding on how proper data preparing is important, even before picking and tuning models.

One area of improvement could be to use different combination of feature using techniques like bagging to get the most important features while simultaneously improving the model performance.

Exploring further MLP & NN algorithms will probably give even better results. By utilizing a more complex and robust Neural Network architecture, we might make the model much better to handle outliers.