Capstone Project Report

Nadima Dwihusna

January 12, 2021

1 Definition

1.1 Project Overview

The Titanic was widely considered as the “unsinkable” ship. However, during its voyage on April 15, 1912, the Titanic sank after hitting an iceberg. Because there were not enough lifeboats, 1502 out of 2224 passengers died. Using the passenger data (e.g. name, age, gender, socio-economic class, etc.) that boarded the Titanic during the voyage, there seems to be a pattern where some groups of people were more likely to survive than others. The main objective of the project is to use machine learning to create a model that predicts which passengers survived the Titanic shipwreck. This project is taken from the Kaggle Competition: “Titanic – Machine Learning from Disaster” (https://www.kaggle.com/c/titanic/overview/description). I am doing this project as this will be my first Kaggle competition and first time building a predictive machine learning model.

1.2 Problem Statement

As stated in the Kaggle Competition (https://www.kaggle.com/c/titanic/overview), the main goal of this project is to build a machine learning model that answers the following question: “What sorts of people were more likely to survive?”. The data of the passengers who boarded the Titanic in 1912 is provided and to be used to build the model that predicts if the passenger survives.

1.3 Evaluation Metrics

An accuracy score and confusion matrix of the classification results will be used as an evaluation metric to quantify the performance of the model.

- The accuracy score compares the total of true positives and true negatives divided by the total number of samples. In other words, this accuracy score indicates how much the machine learning model predicts the outcome of the passenger correctly to the total number of passengers.
- The confusion matrix visualizes the accuracy of the machine learning model by comparing the machine learning model predictions to the actual outcomes. The goal of the project is

to have a high number of true positives and true negatives where the actual outcomes matches with the machine learning predictions.

	1 (Predicted)	0 (Predicted)
1 (Actual)	True Positive	False Negative
0 (Actual)	False Positive	True Negative

Figure 1: Confusion matrix to visualize accuracy of machine learning models.

2 Analysis

2.1 Data Exploration

The training and testing data set can be found in the Kaggle Competition website (https://www.kaggle.com/c/titanic/data). The data is split into two groups:

- **training set (train.csv)**
- **test set (test.csv)**

The training set is used to build the machine learning models. This training set contains the outcomes (ground truth) for each passenger. The test set is used to see how well the model performs with new data. The test set does not have ground truth for each passenger as it is my job to predict the outcomes (passenger survive or not). Additionally, a **gender_submission.csv** file is provided. This file shows an example of what a submission file should look like. All the features in the dataset describes the passenger information as follow:

Feature Variable	Definition	Key
PassengerId (integer)	Passenger ID / count number	
Survived (integer)	Survival	0 = No, 1 = Yes
Pclass (integer)	Ticket class (socio-economic status or SES)	1 = 1st, 2 = 2nd, 3 = 3rd
Name (string)	Passenger name	
Sex (string)	Sex	

Age (integer)	Age in years (Age is fractional if <1. If the age is estimated, it is in the form of xx.5)	
SibSp (integer)	# of siblings / spouses aboard the Titanic	
Parch (integer)	# of parents / children aboard the Titanic	
Ticket (integer)	Ticket number	
Fare (float)	Passenger fare	
Cabin (string)	Cabin number	
Embarked (string)	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

As part of the data preprocessing, the outliers and missing data are detected throughout the training and testing data. The outliers are detected using the Tukey method and are eliminated from the training and testing dataset. Afterwards in the preprocessing step, the null and missing values are detected in the training and testing data set.

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	881.000000	881.000000	881.000000	711.000000	881.000000	881.000000	881.000000
mean	446.713961	0.385925	2.307605	29.731603	0.455165	0.363224	31.121566
std	256.617021	0.487090	0.835055	14.547835	0.871571	0.791839	47.996249
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	226.000000	0.000000	2.000000	20.250000	0.000000	0.000000	7.895800
50%	448.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.000000	1.000000	3.000000	38.000000	1.000000	0.000000	30.500000
max	891.000000	1.000000	3.000000	80.000000	5.000000	6.000000	512.329200

Figure 2: train.describe() is ran to show the summary of the training data after getting rid of outliers.

2.2 Exploratory Visualization

To summarize the relevant features and characteristic of the dataset, various visualizations have been made. From Figure 3, based on the training data, we can see that only 38.59 % of the passengers survived the sinking of the Titanic.

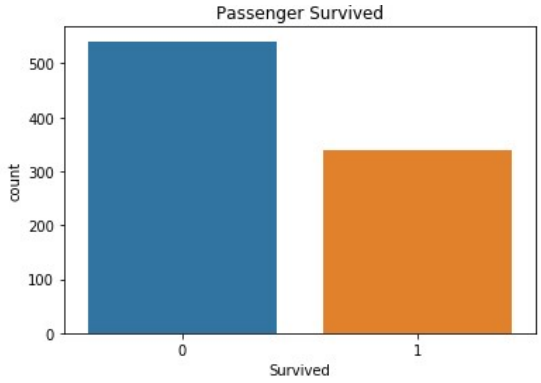

Figure 3: Outcome of passenger in the training data with 1 as survived and 0 as died.

Figure 4 shows a correlation matrix of the different features in the data set. Overall, Parch and SibSp is positively correlated, and Age and Parch seems negatively correlated.

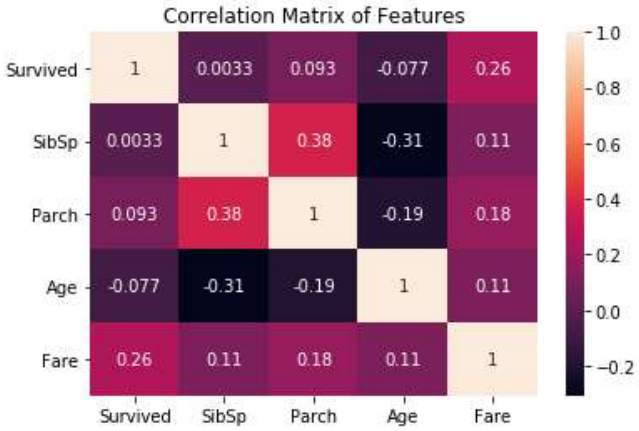

Figure 4: Correlation Matrix of Features and Label in the training data set. The Survived values serve as the label for the training in supervised machine learning.

As seen in Figures 5 and Figure 6, further visualizations have been made for each feature compared to the probability of survival. Seen in these figures, females (74.75% survival probability) are more likely to survive than males. The passengers with Pclass 1 and embarked from C (Cherbourg) are also more likely to survive. Also, seen from the age bell curves in Figure 6, young children / babies are more likely to survive.

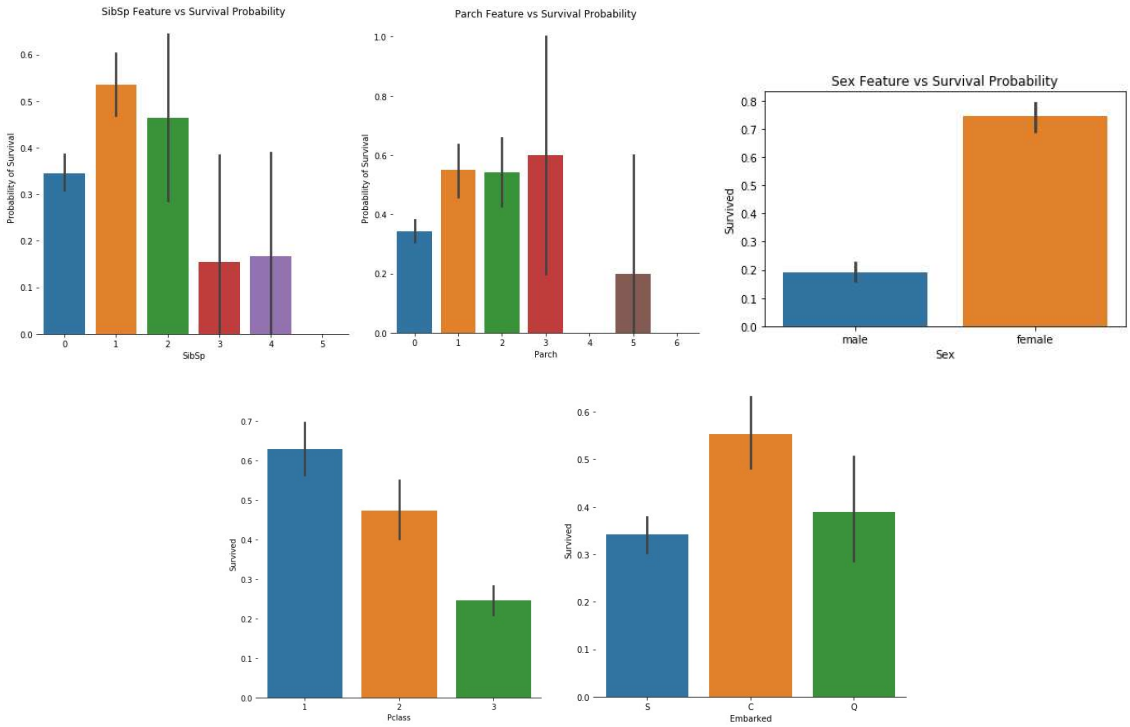

Figure 5: Features versus the probability of survival in the training data.

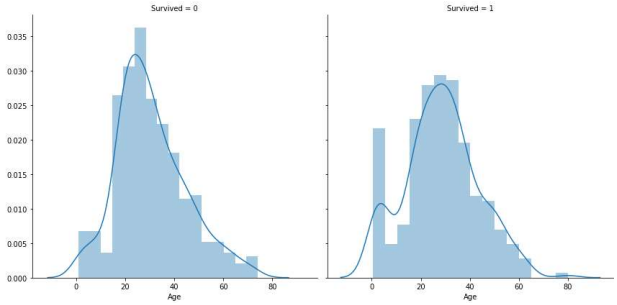

Figure 6: Age group features with bell curve versus the probability of survival in the training data.

2.3 Algorithms and Techniques

Pandas and NumPy libraries were used for data processing and the machine learning model are built using scikit-learn estimators. The quality are measured using the performance metrics or loss function. The Scikit-Learn python open source machine learning library is used for all the processing, modelling, and prediction steps. These are the four machine learning algorithms trained and optimized to predict the passengers' outcomes in the Titanic:

1. Support Vector Classifier (SVC)
2. Decision Tree
3. Random Forest
4. K-Nearest Neighbors (KNN).

I chose SVC because it is known to be a good algorithm for classification or regression problems. SVC transforms the data and finds the optimal boundary between the outputs. I chose decision tree and random forest because these works well in classification problem. It would also be interesting to compare the outcomes since the random forest combines the output of individual decision trees to generate the final output. Lastly, KNN is used since it is an intuitive and simple classification algorithm I have used before in other projects.

2.4 Benchmark

For benchmarking, I decided to compare the four machine learning classifiers and evaluate the mean accuracy of each by a stratified k-fold cross validation process. The k-fold cross validation splits the input into k subsets of data or folds and evaluate each machine learning algorithms. For benchmark, the default “standard” hyperparameters that relates to the domain, problem statement, and solution. Afterwards, the models are be optimized with different hyperparameters. These standard benchmark models are used to objectively compare and evaluate the results with the optimized models. This will help identify improve the final model performance.

3 Methodology

3.1 Data Preprocessing

After visualizing and getting a better understanding of the data and its features, the training and testing data is further processed. As stated in previous section, any outliers are removed to clean the data. Then, I focused on fill in any missing data specifically in Age and Cabin features.

Seen in Figure 7, there are 256 missing values for Age for the whole training and testing dataset. Since age seems to be an important factor to determining the passenger outcomes, I decided to calculate the missing Age values. I filled in the missing age values with median age of similar Pcalss, Parch, and SibSp features. Additionally, there is also a high number of missing Cabin values for the training and testing data. I decided to add an extra cabin ‘N’ for passengers with no assigned cabin. Seen in Figure 8, most of the passengers do not have assigned cabin and that passengers with assigned cabins are more likely to survive the Titanic.

```
train = train.fillna(np.nan)
train.info()
train.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 881 entries, 0 to 880
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  881 non-null    int64
 1   Survived     881 non-null    int64
 2   Pclass       881 non-null    int64
 3   Name         881 non-null    object
 4   Sex          881 non-null    object
 5   Age          711 non-null    float64
 6   SibSp        881 non-null    int64
 7   Parch        881 non-null    int64
 8   Ticket       881 non-null    object
 9   Fare         881 non-null    float64
 10  Cabin        201 non-null    object
 11  Embarked     881 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 82.7+ KB
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            170
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          680
Embarked         0
dtype: int64
```

```
test = test.fillna(np.nan)
test.info()
test.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Pclass       418 non-null    int64
 2   Name         418 non-null    object
 3   Sex          418 non-null    object
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64
 6   Parch        418 non-null    int64
 7   Ticket       418 non-null    object
 8   Fare         418 non-null    float64
 9   Cabin        91 non-null     object
 10  Embarked     418 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          327
Embarked         0
dtype: int64
```

Figure 7: Null and missing values in the training (left) and testing data set (right). In the training data, there are 170 missing values for Age and 680 missing values for Cabin. In the testing data, there are 86 missing values for Age and 327 missing values for Cabin.

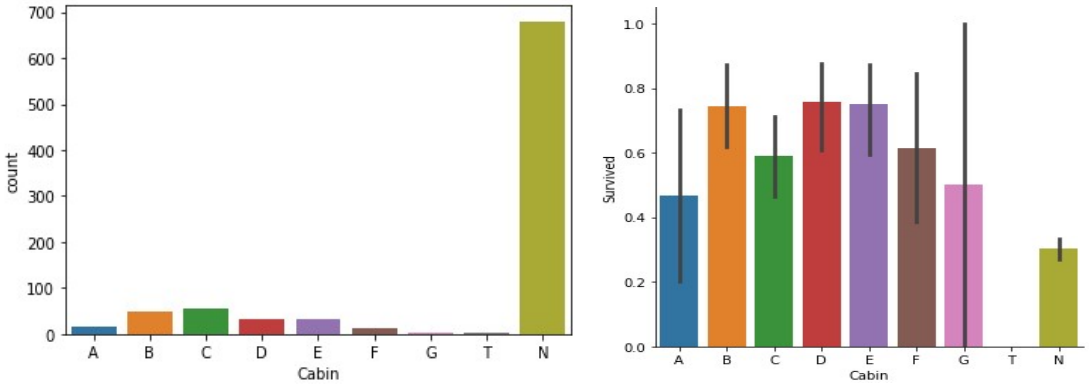

Figure 8: Passenger count based on assigned cabin (‘N’ represents no cabin) in the training data (left). Passenger survival probability based on assigned cabin in the training data (right).

After eliminating outliers and computing missing values, all the features in the training and testing data are converted to integers. The Cabin features are represented by integer 0 to 8, Embarked features are represented by integer 0 to 2, and Pclass features as categorical values.

```
train["Cabin"] = train["Cabin"].map({"A": 0, "B": 1, "C": 2, "D": 3, "E": 4, "F": 5, "G": 6, "T": 7, "N": 8})
```

```
train["Embarked"] = train["Embarked"].map({"S": 0, "C":1, "Q": 2})
```

Then as a feature selection step, the useless features are dropped in the training and testing dataset. I decided to drop PassengerId, Name, and Ticket features because these features do not help the machine learning algorithm predict the final survival outcome. Figure 9 represents the first few rows of the final processed training data with all the features and Survived label.

	Survived	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked	Pc_1	Pc_2	Pc_3
0	0	0	22.0	1	0	7.2500	8	0	0	0	1
1	1	1	38.0	1	0	71.2833	2	1	1	0	0
2	1	1	26.0	0	0	7.9250	8	0	0	0	1
3	1	1	35.0	1	0	53.1000	2	0	1	0	0
4	0	0	35.0	0	0	8.0500	8	0	0	0	1

Figure 9: train.head() is run to show the first few rows of the training data and Survived label.

In the end, after processing, the train.csv training data are then split 75% for training and 25% for validation. The test.csv testing data will be used for the end model evaluation.

```
train["Survived"] = train["Survived"].astype(int)

X = train.drop(labels = ["Survived"], axis=1) #features
Y = train.Survived #labels
```

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.25, random_state=101)
```

3.2 Implementation

As stated previously, for benchmarking, I compared the four different machine learning algorithms (SVC, Decision Tree, Random Forest, and KNN) and evaluate the mean accuracy of each by a stratified k-fold cross validation. Figure 10 shows the four benchmark algorithms using standard hyperparameters. These benchmarks will be compared to the optimized models to see how much improvements have been made.

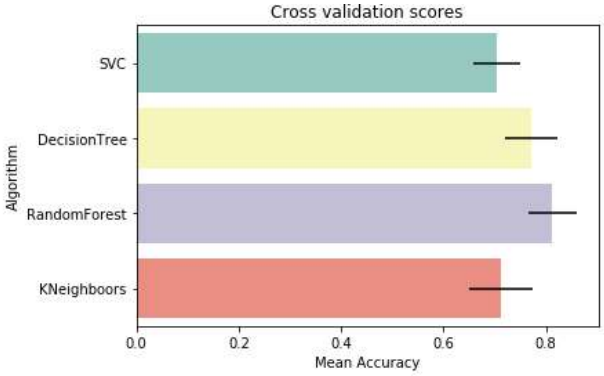

Figure 10: K-Fold Cross validation score for the machine learning models with standard hyperparameters for initial benchmarking. Overall, the mean accuracy scores are similar for all.

For the next steps, each model hyperparameters are tuned and optimized using GridSearch CV class. Each algorithm creates a grid with all the possible hyperparameters and get model with each possibility. This step helps compare the optimized model with the initial benchmark models with standard hyperparameters. The optimized models should have a higher accuracy score than the benchmark models.

```
RFC = RandomForestClassifier()

rf_param_grid = {"max_depth": [None],
                 "max_features": [1, 3, 10],
                 "min_samples_split": [2, 3, 10],
                 "min_samples_leaf": [1, 3, 10],
                 "bootstrap": [False],
                 "n_estimators" :[100,300],
                 "criterion": ["gini"]}


gsRFC = GridSearchCV(RFC,param_grid = rf_param_grid, cv=kfold, scoring="accuracy", n_jobs= 4, verbose = 1)
gsRFC.fit(x_train,y_train)
RFC_best = gsRFC.best_estimator_

# Best score
gsRFC.best_score_
```

After optimizing all four models hyperparameters, I plotted the learning curves to see any effects of overfitting on the training data. Figure 11 shows the learning curves for all four models. From the plots, it is noted that the Decision Tree seems to be overfitting the model where the training score is very high and cross validation score is very low.

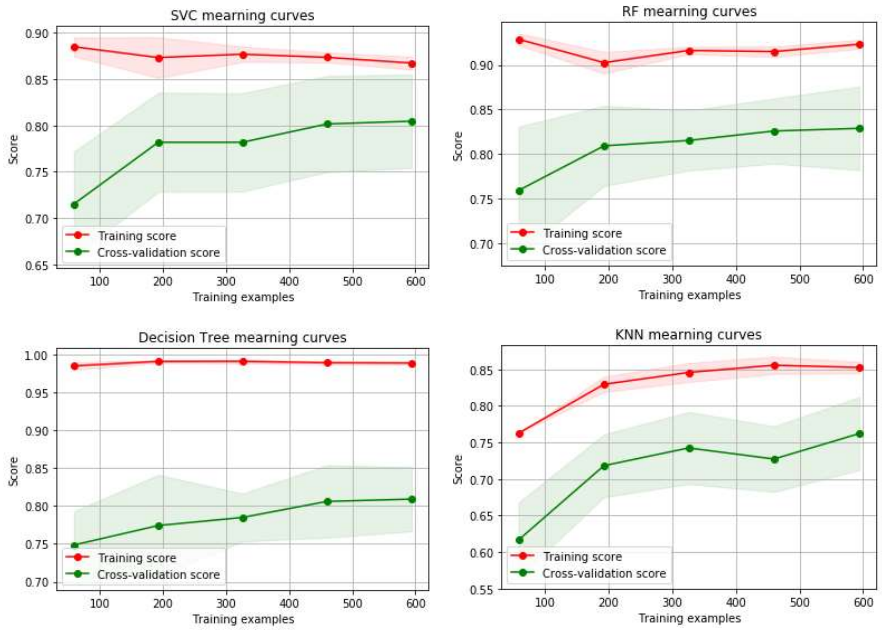

Figure 11: Learning curves for all four machine learning models (SVC, Random Forest, Decision Tree, and K-Nearest Neighbors) to evaluate model performance and see overfitting.

After training the machine learning model, the models are then used to predict the outcome of the passengers in the testing data set. The following images represent the prediction testing results and confusion matrices of the prediction versus actuals.

```
def get_scores(predictor, x_train, y_train):
    print(predictor, ":")
    print()
    print("training: ", predictor.score(x_train, y_train))
    print("testing: ", accuracy_score(predictor.predict(x_test), y_test))
    print()
    print("confusion matrix:")
    print(confusion_matrix(predictor.predict(x_test), y_test))

    print()
    print("classification report:")
    print(classification_report(predictor.predict(x_test), y_test))
    print("------------------------------------------------------------")
```

```
SVC(C=200, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False) :

training:  0.8636363636363636
testing:  0.7647058823529411

confusion matrix:
[[100  23]
 [ 29  69]]

classification report:
              precision    recall  f1-score   support

           0       0.78      0.81      0.79       123
           1       0.75      0.70      0.73        98

    accuracy                           0.76       221
   macro avg       0.76      0.76      0.76       221
weighted avg       0.76      0.76      0.76       221

------------------------------------------------------------
AdaBoostClassifier(algorithm='SAMME.R',
                   base_estimator=DecisionTreeClassifier(ccp_alpha=0.0,
                                                         class_weight=None,
                                                         criterion='entropy',
                                                         max_depth=None,
                                                         max_features=None,
                                                         max_leaf_nodes=None,
                                                         min_impurity_decrease=0.0,
                                                         min_impurity_split=None,
                                                         min_samples_leaf=1,
                                                         min_samples_split=2,
                                                         min_weight_fraction_leaf=0.0,
                                                         presort='deprecated',
                                                         random_state=None,
                                                         splitter='best'),
                   learning_rate=0.3, n_estimators=2, random_state=7) :

training:  0.9878787878787879
testing:  0.8009049773755657

confusion matrix:
[[110  25]
 [ 19  67]]

classification report:
              precision    recall  f1-score   support

           0       0.85      0.81      0.83       135
           1       0.73      0.78      0.75        86

    accuracy                           0.80       221
   macro avg       0.79      0.80      0.79       221
weighted avg       0.80      0.80      0.80       221
```

```
------------------------------------------------------------
RandomForestClassifier(bootstrap=False, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features=3,
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=3, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=300,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False) :

training:  0.9272727272727272
testing:  0.7782805429864253

confusion matrix:
[[108  28]
 [ 21  64]]

classification report:
              precision    recall  f1-score   support

           0       0.84      0.79      0.82       136
           1       0.70      0.75      0.72        85

    accuracy                           0.78       221
   macro avg       0.77      0.77      0.77       221
weighted avg       0.78      0.78      0.78       221

------------------------------------------------------------
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='manhattan',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform') :

training:  0.8545454545454545
testing:  0.7420814479638009

confusion matrix:
[[102  30]
 [ 27  62]]

classification report:
              precision    recall  f1-score   support

           0       0.79      0.77      0.78       132
           1       0.67      0.70      0.69        89

    accuracy                           0.74       221
   macro avg       0.73      0.73      0.73       221
weighted avg       0.74      0.74      0.74       221

------------------------------------------------------------
```

3.3 Refinement

The table below summarizes the accuracy results. Comparing the different results, the Decision Tree Classifier model performs with the highest accuracy of 80.09% followed by the Random Forest model accuracy of 77.82%.

	Accuracy
SVC	0.7647058823529411
Decision Tree	0.8009049773755657
Random Forest	0.7782805429864253
KNN	0.7420814479638009

The model hyperparameters were refined and optimized using the GridSearchCV class as stated previously. Other than optimizing the model hyperparameters, another way to improve the accuracy score is to select the correct features to be used during training. Selecting the correct features is one of the most challenging part about this project. At first, I selected all features in the data set to train the algorithm. For example, I included the Name and Ticket feature. I processed the Name into Title features, where the Title (e.g. Mr., Mrs., Master, Dr., etc.) becomes its own feature. In the end, I decided to drop these PassengerId, Name, Title, and Ticket features as they lower the accuracy score and are irrelevant to the outcome.

From optimizing the hyperparameters and applying the necessary features, I was able to refine and improve the performance accuracy of my machine learning model. I learned that feature selection is as important as model optimization. Feature selection allows the machine learning algorithm to train faster as it reduces the complexity of a model making it easier to interpret.

4 Results

4.1 Model Evaluation and Validation

In the end, the solution is a machine learning model that predicts if the passenger would survive based on the given passenger information and features. The model was tested to predict which passengers survive the Titanic. Comparing the four initial benchmark models, the Random Forest seemed to perform the best followed by the Decision Tree classifier. However, after GridSearchCV and optimizing the hyperparameters, in the end, the Decision Tree Classifier performed with the highest accuracy of 80.09%. The following shows the final Decision Tree hyperparameters:

```
base_estimator=DecisionTreeClassifier(ccp_alpha=0.0,
                                      class_weight=None,
                                      criterion='entropy',
                                      max_depth=None,
                                      max_features=None,
                                      max_leaf_nodes=None,
                                      min_impurity_decrease=0.0,
                                      min_impurity_split=None,
                                      min_samples_leaf=1,
                                      min_samples_split=2,
                                      min_weight_fraction_leaf=0.0,
                                      presort='deprecated',
                                      random_state=None,
                                      splitter='best'),
learning_rate=0.3, n_estimators=2, random_state=7) :
```

4.2 Justification

In the end, the results (models with optimized hyperparameters) were compared to the initial benchmark results (models with standard hyperparameters). The model performance to predict the passenger outcomes in the Titanic significantly improved after optimizing the hyperparameters through GridSearchCV and feature selection.

The results from the Decision Tree classifier with an accuracy of 80.09 % is justifiable. As seen in Figure 12, there are a total of 110 True Negatives and 67 True Positives which are much higher when compared to the Random Forest classifier in Figure 13. This means that the Decision Tree classifier made better prediction of which passenger survives and died in the Titanic.

	Predict 0	Predict 1
Actual 0	110	25
Actual 1	19	67

Figure 12: Decision Tree classifier with accuracy of 80.09 %.

	Predict 0	Predict 1
Actual 0	108	28
Actual 1	21	64

Figure 13: Random Forest classifier with accuracy of 77.82 %.