MSDS 422 Assignment 4b
Husein Adenwala

Introduction

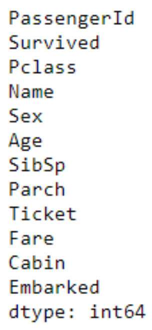
The Purpose of this study is to create a machine learning model that can predict passengers that would survive and passengers that would not survive on the Titanic.   On April 15, 1912, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew (70%).

Data preparation, exploration, visualization

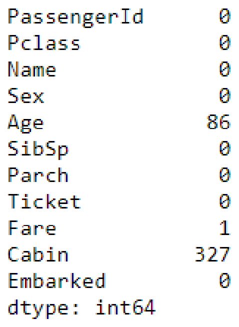
I prepared the dataset for modeling by taking the same steps as in the previous assignment, with some additions. I will outline all the steps below.

The Titanic dataset is split into two datasets: train and test. The train dataset has 891 observations and 12 columns, and the test dataset has 418 observation and does not include the response variable (it therefore has 11 columns). The two datasets did have missing values for Age, Cabin and Fare, as seen in the table below:

<u>Number of missing values for each column</u>



```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```



```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

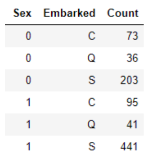
Imputation

I imputed the Age by calculating the median age by grouping the data by Pclass and SibSp (number of siblings/spouse).

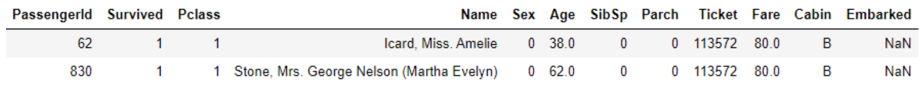
For the missing cabins, I replaced the missing cabin values by a new U class as there was not a pattern for the missing cabin values. It is best to assign as new class U – unknown.

As there was only one missing Fare value, I replaced the value with the mean fare.

For the two missing Embarked values, I replaced the missing values with S as that is the most likely missing value. Both of the passengers in question are women who survived; as seen below, the majority of the women that survived embarked at port S (in the tables, 0 is female, 1 is male).



Sex	Embarked	Count
0	C	73
0	Q	36
0	S	203
1	C	95
1	Q	41
1	S	441

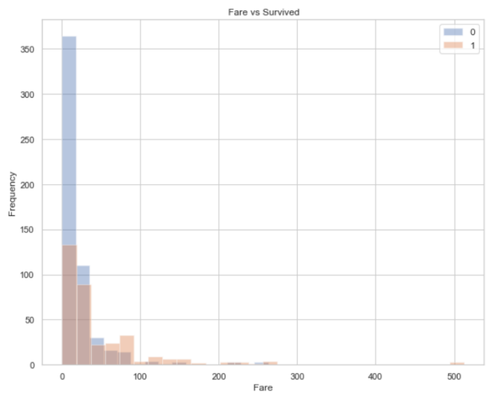


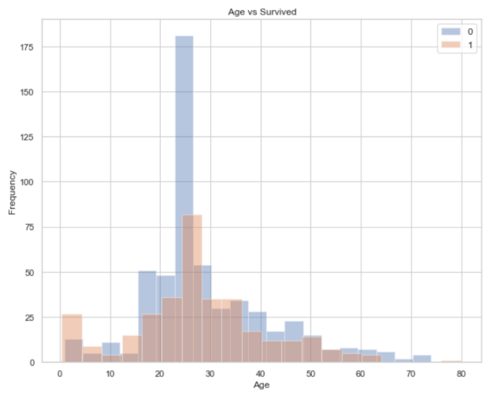
PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
62	1	1	Icard, Miss. Amelie	0	38.0	0	0	113572	80.0	B	NaN
830	1	1	Stone, Mrs. George Nelson (Martha Evelyn)	0	62.0	0	0	113572	80.0	B	NaN

Features Selection

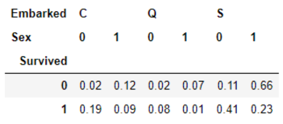
For features selection, I created histograms and cross tables to identify features which have value or categories which discriminate between survived and not survived.

The plot below for Fare and Age shows that passengers that paid a Fare between 0-50 were most likely to not survive and passengers between the age of 16-30 were most likely to not survive (in the graphs below, 0 is not survived and 1 is survived).

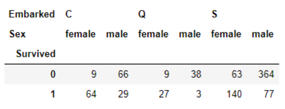




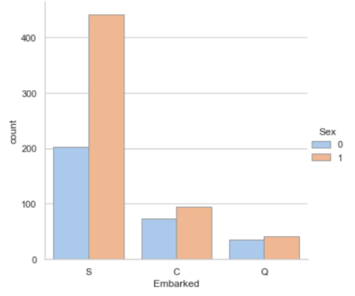
Women that embarked at S were more likely to survive and passengers with unknown cabins were most likely to not survive.

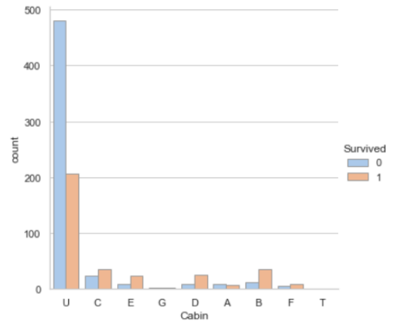


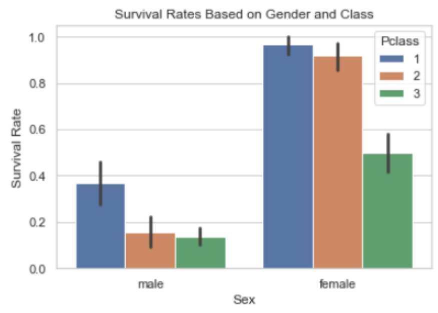
Embarked	C		Q		S	
Sex	0	1	0	1	0	1
Survived						
0	0.02	0.12	0.02	0.07	0.11	0.66
1	0.19	0.09	0.08	0.01	0.41	0.23

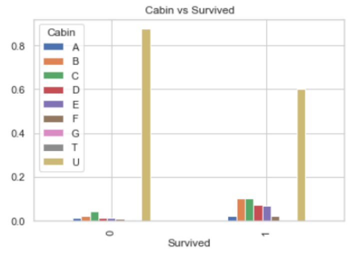


Embarked	C		Q		S	
Sex	female	male	female	male	female	male
Survived						
0	9	66	9	38	63	364
1	64	29	27	3	140	77





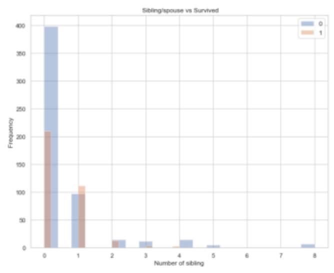


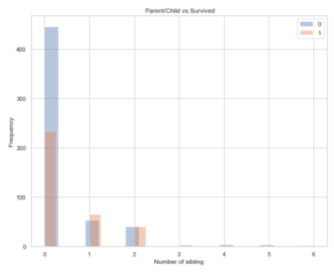


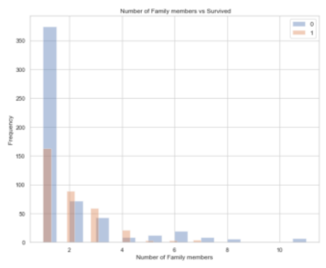
Feature Engineering

Family size

The people without parent/child or sibling/spouse were less likely to survive. I combined the two to create a new feature, family size, that I will use in the model in place of the two features (parent/child and sibling/spouse), that have mild collinearity.

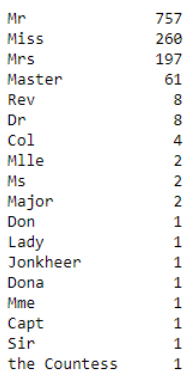




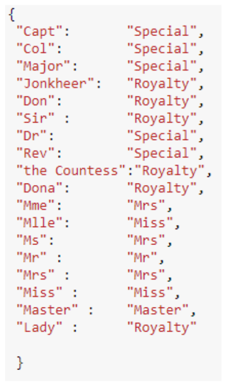


Titles category

The names column has prefixes such as Mr, Mrs , Miss , Major, Col and so on and I have extracted them to create categorical variable called Titles with the following mapping:



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



```
{
 "Capt":       "Special",
 "Col":        "Special",
 "Major":      "Special",
 "Jonkheer":   "Royalty",
 "Don":        "Royalty",
 "Sir" :       "Royalty",
 "Dr":         "Special",
 "Rev":        "Special",
 "the Countess":"Royalty",
 "Dona":       "Royalty",
 "Mme":        "Mrs",
 "Mlle":       "Miss",
 "Ms":         "Mrs",
 "Mr" :        "Mr",
 "Mrs" :       "Mrs",
 "Miss" :      "Miss",
 "Master" :    "Master",
 "Lady" :      "Royalty"

 }
```

Pclass categorical

I converted Pclass to a categorical variable from a numeric variable, which is the appropriate data type for Pclass.

I selected the following eight features—Pclass, Sex, Age, Fare, Cabin, Embarked, Titles and Family size—for my predictive model.

For the categorical data – Embarked, Pclass, Titles and Cabin—I created N-1 dummy variables and the final features that were used are as follows:

Sex	Binary
Age	Numeric
Fare	Numeric
Cabin A	Dummy Binary
Cabin B	Dummy Binary
Cabin C	Dummy Binary
Cabin D	Dummy Binary
Cabin E	Dummy Binary
Cabin F	Dummy Binary
Cabin G	Dummy Binary
Cabin T	Dummy Binary
Embarked S	Dummy Binary
Embarked C	Dummy Binary
Pclass_2	Dummy Binary

Pclass_3	Dummy Binary
Master	Dummy Binary
Miss	Dummy Binary
Mr	Dummy Binary
Mrs	Dummy Binary
Special	Dummy Binary
FamilySize	Numeric

Review research design and modeling methods

For this analysis, I used a classification model and Tree Based model because of the Binary nature of our response variable Survived. Specifically, I used Logistic Regression (LR), Random Forest Classifier (RF) and Extra Trees (ET). Each of these models have various characteristics that make them suitable for specific datasets.

LR is easier to implement, interpret, and very efficient to train for binary data and will be the benchmark model.

RF and ET are ensemble methods of decision tree which are Non-parametric, which means they don't have to worry about outliers or whether the data is linearly separable. Their main disadvantage is that they easily overfit, but that's where ensemble methods like RF and ET that minimize the overfit come in.

All three models seem to be appropriate for the Titanic dataset.

To evaluate these models, I used only the selected 8 features (21 with dummy coded features) and did an 80/20 split on the dataset for Training and Testing respectively. I used the same test and training data on all the models to do a fair comparison.

I also used the k-fold cross validation (k=10) technique to calculate and compare the accuracy for each model. Additionally, I ran a comparison of accuracy score, AUC score, Precision, Recall and F1 score between train and test data as an evaluation metric. I also calculated the OOB Score to compare between the RF and ET models.

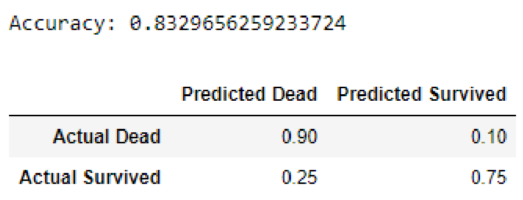
Review results, evaluate models

Logistic Regression:

Logistic Regression performed well on Precision, Recall, Accuracy and f1-score and the performance between test and train data was very similar. With the imbalanced data, where there is a high number of 0 as compared to 1 for the response variable survived; the recall score is important. The Precision and Recall scores were similar, but it had some variation between test and train data. The False positive rate was low and similar in both test and train data.

Adjusting the probability threshold did not have a significant impact on improving model performance.

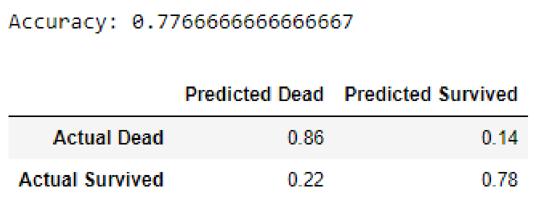
Train



```
Accuracy: 0.8329656259233724
```

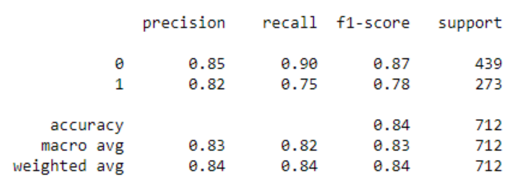
	Predicted Dead	Predicted Survived
Actual Dead	0.90	0.10
Actual Survived	0.25	0.75

Test

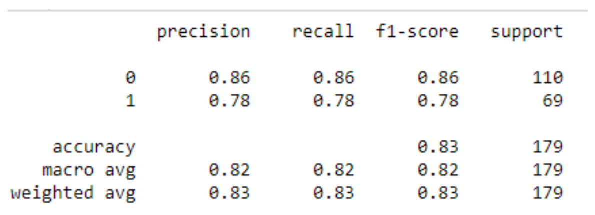


```
Accuracy: 0.7766666666666667
```

	Predicted Dead	Predicted Survived
Actual Dead	0.86	0.14
Actual Survived	0.22	0.78



```
              precision    recall  f1-score   support

           0       0.85      0.90      0.87       439
           1       0.82      0.75      0.78       273

    accuracy                           0.84       712
   macro avg       0.83      0.82      0.83       712
weighted avg       0.84      0.84      0.84       712
```



```
              precision    recall  f1-score   support

           0       0.86      0.86      0.86       110
           1       0.78      0.78      0.78        69

    accuracy                           0.83       179
   macro avg       0.82      0.82      0.82       179
weighted avg       0.83      0.83      0.83       179
```

Below is the equation and interpretation of the logistic regression:

*Y_hat = 1 .46 -0.789*Sex -0.017*Age +0.01*Fare +0.798*Cabin A +0.389*Cabin B +0.265*Cabin C +1.221*Cabin D +1.889* Cabin E +0.59*Cabin F -0.017*Cabin G -0.074*Cabin T +0.025*Embarked S +0.383* Embarked C +0.*Pclass_2 +0.*Pclass_ 3 +.307*Master +0.399*Miss -1.763*Mr +1.04*Mrs -0.531*Special -0.482*Familysize*

The intercept $= 1.46$ can be interpreted as follows when all other coefficients = 0:

- exp(1.46) = 4.3, i.e. we estimate that the log odds of survival increases by 4.3 when sex is Female (0), cabin is U, Embark is Q, Pclass is 1 and Title is Royalty

Sex $= -0.789$ is a binary variable.

- exp(-0.789)= 0.45 , i.e we estimate that the log odds of survival decreases by 0.45 When sex is Male (1)

Age = -0.017 is interpreted as follows when all other coefficients are = 0 :

- exp(-0.017) = 0.98 , i.e we estimate that the log odds of survival decreases by 0.98 with increase in Age

Fare = 0.01 can be interpreted as follows when all other coefficients are = 0 :

- exp(0.01) = 1.003 , i.e we estimate that the log odds of survival increases by 1.003 with increase in Fare

Cabin A= 0.789 can be interpreted as follows when all other coefficients are = 0 :

- exp(0.789) = 2.2 , i.e we estimate that the log odds of survival increases by 2.2 when cabin is A

Cabin B= 0.389 can be interpreted as follows when all other coefficients are = 0 :

- exp(0.389) = 1.47 , i.e we estimate that the log odds of survival increases by 1.47 when cabin is B

Cabin C= 0.265 can be interpreted as follows when all other coefficients are = 0 :

- exp(0.265) = 1.3 , i.e we estimate that the log odds of survival increases by 1.3 when cabin is C

Cabin D= 1.221 can be interpreted as follows when all other coefficients are = 0 :

- exp(1.221) = 3.39 , i.e we estimate that the log odds of survival increases by 3.39 when cabin is D

Cabin E= 1.889 can be interpreted as follows when all other coefficients are = 0 :

- exp(1.889) = 6.67, i.e we estimate that the log odds of survival increases by 6.67 when cabin is E

Cabin F = 0.59 can be interpreted as follows when all other coefficients are = 0 :

- exp(0.59) = 1.8 , i.e we estimate that the log odds of survival increases by 1.8 when cabin is F

Cabin G= -0.017 can be interpreted as follows when all other coefficients are = 0 :

- exp(-0.017) = 0.98 , i.e we estimate that the log odds of survival decreases by 0.98 when cabin is G

Cabin T= -0.074 can be interpreted as follows when all other coefficients are = 0 :

- exp(-0.074) = 0.9 , i.e we estimate that the log odds of survival decreases by 0.9 when cabin is T

Embared C= 0.383 can be interpreted as follows when all other coefficients are = 0 :

- exp(0.383) = 1.46 , i.e we estimate that the log odds of survival increases by 1.46 when Embarked is C

Embared S= 0.025 can be interpreted as follows when all other coefficients are = 0 :

- exp(0.025) = 1.02 , i.e we estimate that the log odds of survival increases by 1.02 when Embarked is S

Pclass 2 and Pclass 3 Coefficient were extremely small and had almost no impact on the model.

Family Size= -0.482 can be interpreted as follows when all other coefficients are = 0 :

- exp(-0.482) = 0.62 , i.e we estimate that the log odds of survival decreases by 0.62 when family size increases by 1 unit

Random Forest

RF also performed well on Precision, Recall Accuracy and F1-score and the performance between test and train data was very similar. Like with the Logistic Regression, with the imbalanced data, where there is a high number of 0 as compared to 1 for the response variable survived; the recall score is important. RF has similar recall and precision score and has lower False Positives on train as compared to the test data but it is comparable to Logistic Regression.

Train

```
Accuracy: 0.811878262582488
```

	Predicted Dead	Predicted Survived
Actual Dead	0.94	0.06
Actual Survived	0.19	0.81

```
              precision    recall  f1-score   support

           0       0.89      0.94      0.91       439
           1       0.89      0.81      0.85       273

    accuracy                           0.89       712
   macro avg       0.89      0.88      0.88       712
weighted avg       0.89      0.89      0.89       712
```

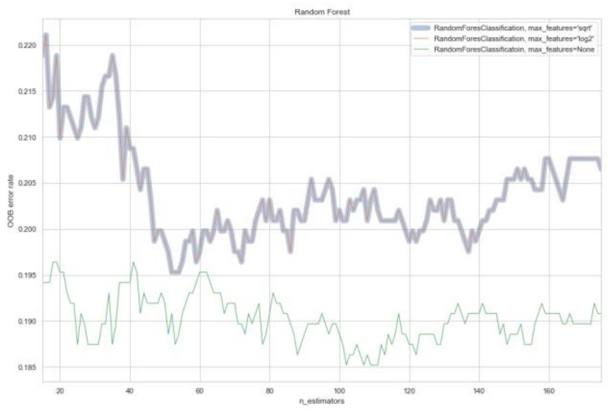
Test

```
Accuracy: 0.8206349206349206
```

	Predicted Dead	Predicted Survived
Actual Dead	0.88	0.12
Actual Survived	0.25	0.75

```
              precision    recall  f1-score   support

           0       0.85      0.88      0.87       110
           1       0.80      0.75      0.78        69

    accuracy                           0.83       179
   macro avg       0.83      0.82      0.82       179
weighted avg       0.83      0.83      0.83       179
```

The graph shows the relationship between the number of estimators and the OOB error:



```
Randreg_oob_score
```

```
0.8146067415730337
```

Feature importance

```
'Sex - 0.154',
'Age - 0.083',
'Fare - 0.097',
'Cabin A - 0.008',
'Cabin B - 0.023',
'Cabin C - 0.016',
'Cabin D - 0.016',
'Cabin E - 0.04',
'Cabin F - 0.008',
'Cabin G - 0.002',
'Cabin T - 0.0',
'Embarked S - 0.03',
'Embarked C - 0.027'
'Pclass_2 - 0.0',
'Pclass_3 - 0.0',
'Master - 0.016',
'Miss - 0.07',
'Mr - 0.237',
'Mrs - 0.071',
'Special - 0.009',
'FamilySize - 0.091'
```

Extra Trees

ET performed best on Precision, Recall, Accuracy and F1-score and the performance between test and train data was very similar. Like the other models, with the imbalanced data, where there is a high number of 0 as compared to 1 for the response variable survives, the precision recall curve and recall score is important. ET has a similar recall score and precision score and has slightly lower false positives on the train and test data as compared to LR and RF.

Train

```
Accuracy: 0.8273515217177188
```

	Predicted Dead	Predicted Survived
Actual Dead	0.91	0.09
Actual Survived	0.18	0.82

```
              precision    recall  f1-score   support

           0       0.89      0.91      0.90       439
           1       0.86      0.82      0.84       273

    accuracy                           0.88       712
   macro avg       0.87      0.87      0.87       712
weighted avg       0.88      0.88      0.88       712
```

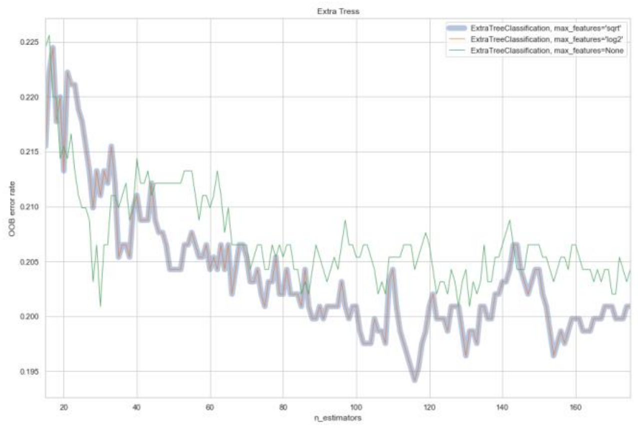
Test

```
Accuracy: 0.8153968253968253
```

	Predicted Dead	Predicted Survived
Actual Dead	0.85	0.15
Actual Survived	0.22	0.78

```
              precision    recall  f1-score   support

           0       0.86      0.85      0.85       110
           1       0.76      0.78      0.77        69

    accuracy                           0.82       179
   macro avg       0.81      0.81      0.81       179
weighted avg       0.82      0.82      0.82       179
```

The graph shows the relationship between the number of estimators and the OOB error:



```
ETreg_obb_score
```

```
0.8188202247191011
```

Feature Importance

```
'Sex - 0.116',
'Age - 0.171',
'Fare - 0.233',
'Cabin A - 0.005',
'Cabin B - 0.014',
'Cabin C - 0.012',
'Cabin D - 0.014',
'Cabin E - 0.025',
'Cabin F - 0.005',
'Cabin G - 0.003',
'Cabin T - 0.001',
'Embarked S - 0.018',
'Embarked C - 0.015',
'Pclass_2 - 0.0',
'Pclass_3 - 0.0',
'Master - 0.011',
'Miss - 0.044',
'Mr - 0.166',
'Mrs - 0.032',
'Special - 0.006',
'FamilySize - 0.109']
```

Summary of three models with training and test data

This summary shows that Extra Trees performed highest in accuracy and AUC on the Test data, followed by Random Forest and finally Logistic regression but all three models are close in performance. The difference in test and train accuracy for Logistic Regression shows overfitting.

Training data

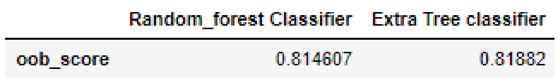
Classifier	Accuracy	AUC
Logistic Regression	0.832966	0.869968
Random Forrest Classifier	0.813306	0.965239
Extra Trees Classifier	0.825933	0.926085

Test data

Classifier	Accuracy	AUC
Logistic Regression	0.776667	0.861265
Random Forrest Classifier	0.820635	0.884190
Extra Trees Classifier	0.826667	0.882279

OOB Score and Error

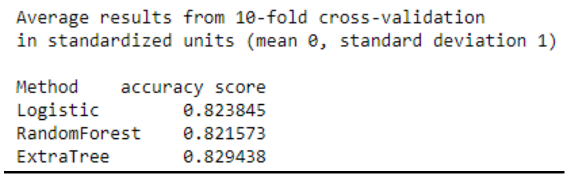
The OOB score is equivalent to the accuracy, which shows that the Extra Trees performs slightly better than the Random Forest.



	Random_forest Classifier	Extra Tree classifier
oob_score	0.814607	0.81882

K-fold cross validation score

Like the test and train performance metrics, Extra Trees performed better than the other two models in K-fold comparison.



```
Average results from 10-fold cross-validation
in standardized units (mean 0, standard deviation 1)

Method      accuracy score
Logistic          0.823845
RandomForest      0.821573
ExtraTree         0.829438
```

Implementation and programming

Using Python Pandas package, I loaded the Titanic Train and test csv file to the Data Frame object.

Exploratory data analysis was done utilizing the function in the Pandas Numpy, seaborn and matplotlib packages as well as methods from ExtraTreesClassifier package in scikit-learn.

Imputation and feature engineering was done by using pandas and Numpy packages.
I then created dummy coded variables utilizing manual created functions, python and pandas methods.

I then created a Numpy array (y) for response variable Survived and removed it from the Titanic Data Frame.

I dropped the unnecessary columns from the Titanic data frame.

At this point, I created an 80/20 train and test split of the Numpy arrays (Y and features), then used the Logistic Regression function in scikit learn package to create a logistic regression model and then evaluated accuracy, ROC, AUC, F1 Score, and Precision for train and test data using scikit learn method and manually created functions.
I used the Random Forest and Extra Trees Classifier from scikit learn to create the Random Forest and Extra Tree classifier model and then evaluated accuracy, ROC, AUC , F1 Score, Precision for train and test data using scikit learn method and manually created functions.

I used hyper-parameter tuning for all the models and used it to optimize all three models. I found that it has significant impact in improving the RF and ET models where the logistic regression had very modest improvement.

Finally, I did K-Fold cross-validation (k=10) using the K-Fold package in scikit-learn. To do this, I created a numpy array of all the data for all the variables and calculated the mean Accuracy of the 10 folds for comparison.

I also calculated the OOB score to compare RF and ET models.

Exposition, problem description, and management recommendations

After evaluating all three models, I would recommend the Extra Trees model. All models are very similar in performance, but based on the following analysis, I feel that Extra Tree is the better fit for this dataset.

Accuracy, AUC, Recall, Precision and F1 score are similar for all the models. The difference between accuracy and AUC for train and test data does not show any significant overfitting and, in fact, Logistic regression has lower accuracy in test that shows some overfitting. However, K-fold validation (k=10) is a better technique to evaluate the model fit with a small dataset. Based on the accuracy score calculated by cross validation, I found that accuracy score was highest for ET, followed by LR and then RF. All of them are within one percent difference.

The logistic regression does well when there is clear linear separation and Random Forest and Extra Trees are non-parametric algorithms, which means that they do not make any assumptions on the underlying data distribution. As we have included all the features in the model (some that have very little impact on the model) it created noise in the data and Extra Tree generally performs better than Random Forest when there is noise in the data due to randomly choosing the split. Extra Tree or Random Forest is very simple to implement as it requires less data preprocessing. Logistic regression is sensitive to dummy coded variables whereas tree-based models can handle categorical data (numeric categorical in python).

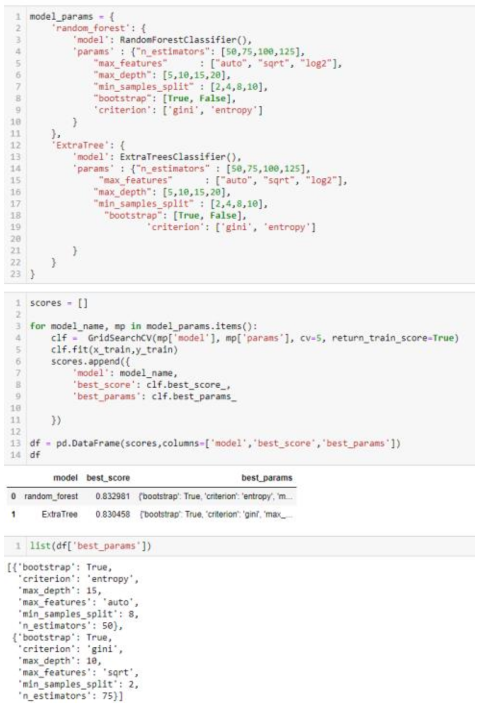
The Titanic data has features that have linear separation but not for all features. Also we did data preprocessing to create new features which made significant impact on logistic regression, but Extra Tree (and RF) performance was similar before and after new features creation which makes it easy to implement and the most suitable model for the Titanic data set.

Appendix

OOB error vs accuracy

```python
# Map a classifier name to a list of (<n_estimators>, <error rate>) pairs.

from collections import OrderedDict
from sklearn.datasets import make_classification
from sklearn.ensemble import RandomForestClassifier

# Author: Kian Ho <hui.kian.ho@gmail.com>
#         Gilles Louppe <g.louppe@gmail.com>
#         Andreas Mueller <amueller@ais.uni-bonn.de>
#
# License: BSD 3 Clause

print(__doc__)

RANDOM_STATE = 123

#fig size
fig_dims = (15, 10)
fig, ax = plt.subplots(figsize=fig_dims)


# NOTE: Setting the `warm_start` construction parameter to `True` disables
# support for parallelized ensembles but is necessary for tracking the OOB
# error trajectory during training.
ensemble_clfs = [
    ("ExtraTreeClassification, max_features='sqrt'",
       ExtraTreesClassifier(warm_start=True, oob_score=True,
                               max_features="sqrt",
                               random_state=RANDOM_STATE,bootstrap=True)),
    ("ExtraTreeClassification, max_features='log2'",
         ExtraTreesClassifier(warm_start=True, max_features='log2',
                               oob_score=True,
                               random_state=RANDOM_STATE,bootstrap=True)),
    ("ExtraTreeClassification, max_features=None",
         ExtraTreesClassifier(warm_start=True, max_features= None,
                               oob_score=True,
                               random_state=RANDOM_STATE,bootstrap=True))
]

# Map a classifier name to a list of (<n_estimators>, <error rate>) pairs.
error_rate = OrderedDict((label, []) for label, _ in ensemble_clfs)

# Range of `n_estimators` values to explore.
min_estimators = 15
max_estimators = 175

for label, clf in ensemble_clfs:
    for i in range(min_estimators, max_estimators + 1):
        clf.set_params(n_estimators=i)
        clf.fit(x, y)

        # Record the OOB error for each `n_estimators=i` setting.
        oob_error = 1 - clf.oob_score_
        error_rate[label].append((i, oob_error))

# Generate the "OOB error rate" vs. "n_estimators" plot.
for label, clf_err in error_rate.items():
    al= 1
    ln = 1
    if label == "ExtraTreeClassification, max_features='sqrt'":
        ln = 8.0
        al = 0.4

    xs, ys = zip(*clf_err)
    plt.plot(xs, ys, label=label, alpha= al, linewidth = ln)




plt.xlim(min_estimators, max_estimators)
plt.xlabel("n_estimators")
plt.ylabel("OOB error rate")
plt.legend(loc="upper right")
plt.title("Extra Tress")
plt.show()
```

Hyperparameter tuning with Grid search



```python
model_params = {
    'random_forest': {
        'model': RandomForestClassifier(),
        'params' : {"n_estimators": [50,75,100,125],
            "max_features"      : ["auto", "sqrt", "log2"],
            "max_depth": [5,10,15,20],
            "min_samples_split" : [2,4,8,10],
            "bootstrap": [True, False],
            'criterion': ['gini', 'entropy']
        }
    },
    'ExtraTree': {
        'model': ExtraTreesClassifier(),
        'params' : {"n_estimators" : [50,75,100,125],
             "max_features"      : ["auto", "sqrt", "log2"],
            "max_depth": [5,10,15,20],
            "min_samples_split" : [2,4,8,10],
              "bootstrap": [True, False],
                      'criterion': ['gini', 'entropy']

        }
    }
}
```

```python
scores = []

for model_name, mp in model_params.items():
    clf =  GridSearchCV(mp['model'], mp['params'], cv=5, return_train_score=True)
    clf.fit(x_train,y_train)
    scores.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_

    })

df = pd.DataFrame(scores,columns=['model','best_score','best_params'])
df
```

	model	best_score	best_params
0	random_forest	0.832981	{'bootstrap': True, 'criterion': 'entropy', 'm...
1	ExtraTree	0.830458	{'bootstrap': True, 'criterion': 'gini', 'max_...

```python
list(df['best_params'])
```

```
[{'bootstrap': True,
  'criterion': 'entropy',
  'max_depth': 15,
  'max_features': 'auto',
  'min_samples_split': 8,
  'n_estimators': 50},
 {'bootstrap': True,
  'criterion': 'gini',
  'max_depth': 10,
  'max_features': 'sqrt',
  'min_samples_split': 2,
  'n_estimators': 75}]
```