TITANIC – DATA SCIENCE SOLUTION

INTRODUCTION

The sinking of the Titanic is one of the most infamous shipwrecks in history.

On April 15, 1912, during her maiden voyage, the widely considered “unsinkable” RMS Titanic sank after colliding with an iceberg. Unfortunately, there were not enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew.

While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others.

In this report, I am to build a predictive model that answers the question: “what sorts of people were more likely to survive?” using passenger data (ie. name, age, gender, socio-economic class, etc) from the ‘**train.csv**’ dataset and to predict the survival status of passengers in the ‘**test.csv**’ dataset.

DATA

Train.csv:

```
train_data.head(10)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

Train.csv contains the details of a subset of the passengers on board (891 to be exact) and importantly, reveals whether they survived or not, also known as the “ground truth”.

Test.csv:

```
test_data.head(10)
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
5	897	3	Svensson, Mr. Johan Cervin	male	14.0	0	0	7538	9.2250	NaN	S
6	898	3	Connolly, Miss. Kate	female	30.0	0	0	330972	7.6292	NaN	Q
7	899	2	Caldwell, Mr. Albert Francis	male	26.0	1	1	248738	29.0000	NaN	S
8	900	3	Abrahim, Mrs. Joseph (Sophie Halaut Easu)	female	18.0	0	0	2657	7.2292	NaN	C
9	901	3	Davies, Mr. John Samuel	male	21.0	2	0	A/4 48871	24.1500	NaN	S

The `test.csv` dataset contains similar information but does not disclose the “ground truth” for each passenger. Using the patterns I find in the train.csv data, predict whether the other 418 passengers on board found in ‘test.csv’ survived.

METHOD

Using Machine Learning in python, I first and foremost import my libraries: numpy, matplotlib.pyplot, seaborn, pandas, RandomForestClassifier.

Acquire data

The Python Pandas packages helps us work with our datasets. We start by acquiring the training datasets into Pandas DataFrames.

```
train_data_path = (r'C:\Users\samha\Documents\Samuel_ZroNet\Me\python\Jupyter files\Titanic Survival Project\ML-titanic-survival
train_data = pd.read_csv(train_data_path)
```

Analyze by describing data

Pandas also helps describe the datasets answering following questions early in our project.

Which features are available in the dataset?

```
train_data.columns
```

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'embarked'],
      dtype='object')
```

Which features are categorical?

These values classify the samples into sets of similar samples. Within categorical features are the values nominal, ordinal, ratio, or interval based? Among other things this helps us select the appropriate plots for visualization.

Categorical: Survived, Sex, and Embarked. **Ordinal**: Pclass.

Which features are numerical?

Which features are numerical? These values change from sample to sample. Within numerical features are the values discrete, continuous, or timeseries based? Among other things this helps us select the appropriate plots for visualization.

Continuous: Age, Fare. **Discrete**: SibSp, Parch.

Preview of the data

```
In [88]: train_data.head(10)
```

Out[88]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

Which features are mixed data types?

Numerical, alphanumeric data within same feature. These are candidates for correcting goal.

- Ticket is a mix of numeric and alphanumeric data types. Cabin is alphanumeric.

Which features may contain errors or typos?

This is harder to review for a large dataset, however reviewing a few samples from a smaller dataset may just tell us outright, which features may require correcting.

- Name feature may contain errors or typos as there are several ways used to describe a name including titles, round brackets, and quotes used for alternative or short names.

Which features contain blank, null or empty values?

These will require correcting.

- Cabin > Age > Embarked features contain a number of null values in that order for the training dataset.
- Cabin > Age are incomplete in case of test dataset.

What are the data types for various features?

Helping us during converting goal.

- Seven features are integer or floats. Six in case of test dataset.
- Five features are strings (object).

What is the distribution of categorical features?

- Names are unique across the dataset (count=unique=891)
- Sex variable as two possible values with 65% male (top=male, frequency =577/count=891).
- Cabin values have several duplicates across samples. Alternatively several passengers shared a cabin.
- Embarked takes three possible values. S port used by most passengers (top=S)
- Ticket feature has high ratio (22%) of duplicate values (unique=681).

Benchmark file which contains the initial predictions to beat predicted the survival of all females and the death of all males. With this, lets use the train data to ascertain this claim.

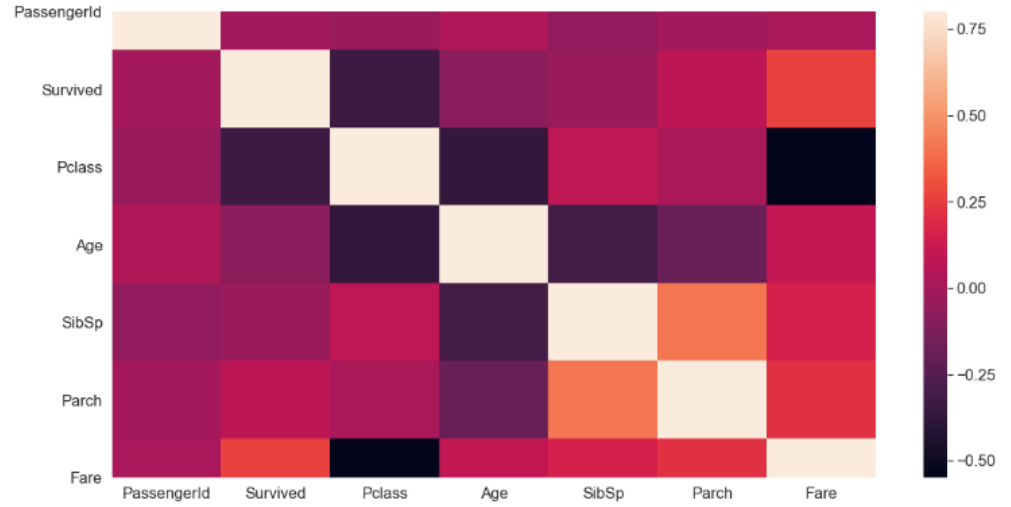
```
In [91]: #benchmark submission file predicted the survival of females and the death of females

In [92]: #so let's check
         women = train_data.loc[train_data.Sex == 'female']["Survived"]
         rate_women = sum(women)/len(women)

         print("Percentage of women who survived:", rate_women)

         Percentage of women who survived: 0.7420382165605095

In [93]: men = train_data.loc[train_data.Sex == 'male']["Survived"]
         rate_men = sum(men)/len(men)

         print("% of men who survived:", rate_men)

         % of men who survived: 0.18890814558058924
```

From the above analysis, about 74% of Women survived and about 19% of Men survived.

Correlation Matrix

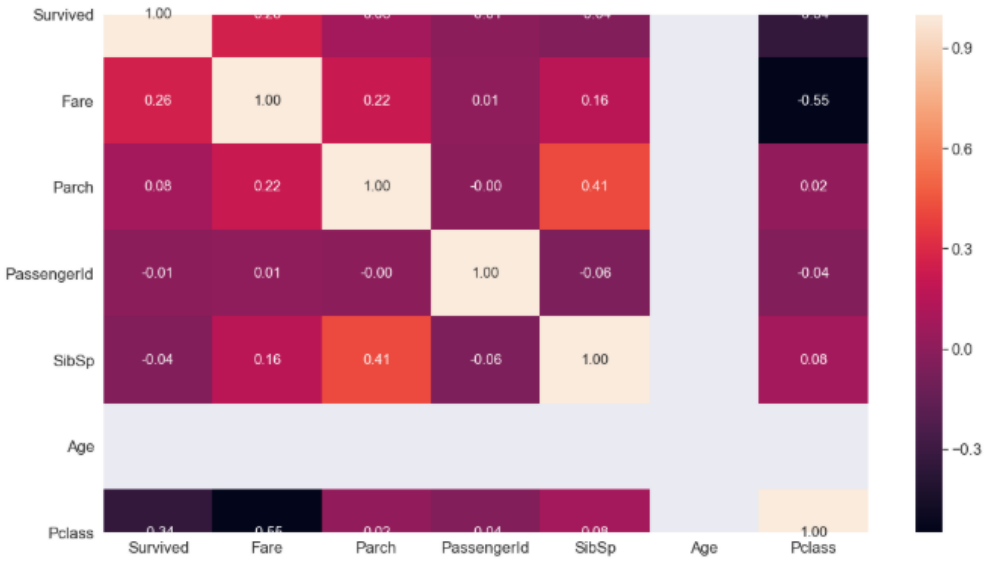
Using seaborn, we can create a correlation matrix to identify the correlation between the variables. Most importantly between the target variable (Survived) and the independent variables (features)

```
In [95]: corrmat = train_data.corr()
         f, ax = plt.subplots(figsize=(18, 9))
         sns.heatmap(corrmat, vmax=.8, square=False);
```



Taking a closer look:

```
In [96]: #taking a closer look
         #'Survived' correlation matrix
         k = 7 #number of variables for heatmap
         cols = corrmat.nlargest(k, 'Survived')['Survived'].index
         cm = np.corrcoef(train_data[cols].values.T)
         sns.set(font_scale=1.5)
         plt.subplots(figsize=(18,10))
         hm = sns.heatmap(cm, cbar=True, annot=True, square=False, fmt='.2f', annot_kws={'size': 15}, yticklabels=cols.values, xticklabel
         plt.show();
```



Checking for Missing Values

To avoid any encounter with Nan values in our classification problem, we can identify and delete all null values for a good fit. The code below identifies the features with the most missing values in a descending order.

In [97]:
```
#dealing with missing values
total = train_data.isnull().sum().sort_values(ascending=False)
percent = (train_data.isnull().sum()/train_data.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(10)
```

Out[97]:

	Total	Percent
Cabin	687	0.771044
Age	177	0.198653
embarked	2	0.002245
Fare	0	0.000000
Ticket	0	0.000000
Parch	0	0.000000
SibSp	0	0.000000
Sex	0	0.000000
Name	0	0.000000
Pclass	0	0.000000

Deleting null values

In [98]:
```
train_data = train_data.drop((missing_data[missing_data['Total'] > 1]).index,1)
train_data.isnull().sum().max() #just checking that there's no data missing...
```

Out[98]: 0

Model, predict and solve

Now we are ready to train a model and predict the required solution. There are 60+ predictive modelling algorithms to choose from. We must understand the type of problem and solution requirement to narrow down to a select few models which we can evaluate. Our problem is a classification and regression problem. We want to identify relationship between output (Survived or not) with other variables or features (Gender, Age, Port...). We are also perfoming a category of machine learning which is called supervised learning as we are training our model with a given dataset. With these two criteria - Supervised Learning plus Classification and Regression, we can narrow down our choice of models to a few. These include:

- Logistic Regression
- KNN or k-Nearest Neighbors
- Support Vector Machines
- Naive Bayes classifier
- Decision Tree
- Random Forrest
- Perceptron
- Artificial neural network
- RVM or Relevance Vector Machine

Random Forests is one of the most popular. Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees (n_estimators=150) at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

The model confidence score is the highest among models evaluated so far. We decide to use this model's output (Predictions) for creating our competition submission of results.

In [99]:
```
from sklearn.ensemble import RandomForestClassifier
y = train_data["Survived"]

#correlated features= Fare, Parch
#in decreasing MAE, add Sex

features = ["Fare", "Pclass", "Sex", "Parch"]
X = pd.get_dummies(train_data[features])
X.head(20)
```

Out[99]:

	Fare	Pclass	Parch	Sex_female	Sex_male
0	7.2500	3	0	0	1
1	71.2833	1	0	1	0
2	7.9250	3	0	1	0
3	53.1000	1	0	1	0
4	8.0500	3	0	0	1
5	8.4583	3	0	0	1
6	51.8625	1	0	0	1
7	21.0750	3	1	0	1
8	11.1333	3	2	1	0
9	30.0708	2	0	1	0
10	16.7000	3	1	1	0
11	26.5500	1	0	1	0
12	8.0500	3	0	0	1
13	31.2750	3	5	0	1
14	7.8542	3	0	1	0
15	16.0000	2	0	1	0
16	29.1250	3	1	0	1
17	13.0000	2	0	0	1
18	18.0000	3	0	1	0
19	7.2250	3	0	1	0

Selected features are:

- Fare
- PClass
- Sex
- Parch

Before mainstream predictions, we can divide our train data with the “train_test_split” library so that we can evaluate our predictions with the **Mean Absolute Error.**

In [100]:
```
from sklearn.model_selection import train_test_split
train_X, val_X, train_y, val_y = train_test_split(X, y, random_state=1)
```

In [101]:
```
model = RandomForestClassifier(n_estimators=150, max_depth=5, random_state=1)
model.fit(train_X, train_y)
predictions = model.predict(val_X)
```

In [102]:
```
from sklearn.metrics import mean_absolute_error
print(mean_absolute_error(val_y, predictions))
```

```
0.19282511210762332
```

With constant changes in our features, we can settle on a prediction with a mean absolute error of **0.19.**

Now we can go ahead with our mainstream predictions with test data.

```
In [103]: test_data_path =(r'C:\Users\samha\Documents\Samuel_ZroNet\Me\python\Jupyter files\Titanic Survival Project\ML-titanic-survival-p
          test_data = pd.read_csv(test_data_path)
```

```
In [104]: test_data.head(10)
```

Out[104]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
5	897	3	Svensson, Mr. Johan Cervin	male	14.0	0	0	7538	9.2250	NaN	S
6	898	3	Connolly, Miss. Kate	female	30.0	0	0	330972	7.6292	NaN	Q
7	899	2	Caldwell, Mr. Albert Francis	male	26.0	1	1	248738	29.0000	NaN	S
8	900	3	Abrahim, Mrs. Joseph (Sophie Halaut Easu)	female	18.0	0	0	2657	7.2292	NaN	C
9	901	3	Davies, Mr. John Samuel	male	21.0	2	0	A/4 48871	24.1500	NaN	S

To make sure, our test data is perfect for our classification and void of any NaN values, let us run some descriptions.

```
In [105]: test_data.describe()
```

Out[105]:

	PassengerId	Pclass	Age	SibSp	Parch	Fare
count	418.000000	418.000000	332.000000	418.000000	418.000000	417.000000
mean	1100.500000	2.265550	30.272590	0.447368	0.392344	35.627188
std	120.810458	0.841838	14.181209	0.896760	0.981429	55.907576
min	892.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	996.250000	1.000000	21.000000	0.000000	0.000000	7.895800
50%	1100.500000	3.000000	27.000000	0.000000	0.000000	14.454200
75%	1204.750000	3.000000	39.000000	1.000000	0.000000	31.500000
max	1309.000000	3.000000	76.000000	8.000000	9.000000	512.329200

```
In [106]: #dealing with missing values
          total = test_data.isnull().sum().sort_values(ascending=False)
          percent = (test_data.isnull().sum()/test_data.isnull().count()).sort_values(ascending=False)
          missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
          missing_data.head(10)
```

Out[106]:

	Total	Percent
Cabin	327	0.782297
Age	86	0.205742
Fare	1	0.002392
Embarked	0	0.000000
Ticket	0	0.000000
Parch	0	0.000000
SibSp	0	0.000000
Sex	0	0.000000
Name	0	0.000000
Pclass	0	0.000000

From the above description and null value checks, we can see one of our features in use contains a null value. To deal with this, we can replace the null value with the data mean.

```
In [108]: test_data.mean()

Out[108]: PassengerId    1100.500000
          Pclass            2.265550
          Age              30.272590
          SibSp             0.447368
          Parch             0.392344
          Fare             35.627188
          dtype: float64

In [109]: test_data['Fare'].fillna((test_data['Fare'].mean()), inplace=True)
```

After replacing the null value, we can then predict the Survival of our 418 passengers in the test data.

```
In [110]: X_test = pd.get_dummies(test_data[features])

          model = RandomForestClassifier(n_estimators=150, max_depth=5, random_state=1)
          model.fit(X, y)
          predictions = model.predict(X_test)
          print(predictions)

          [0 1 0 0 1 0 1 0 1 0 0 0 1 0 1 1 0 0 1 1 0 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0
           0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 0 1 1 0 1 0
           1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0
           1 1 1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
           0 0 1 0 0 1 0 0 1 1 0 1 1 0 1 0 0 1 0 0 1 0 0 0 0 0 0 1 1 0 1 1 0 0 1 0 1
           0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 1 0 1 0 0 0 0 1 1 0 1 0 1 0 1 0
           1 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 1
           0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0
           1 0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0
           1 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 1 0 0 0 1 0 1 0 0 1 0 1 1 0 1 0 0 0 1 0
           0 1 0 0 1 1 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0
           0 1 1 1 1 1 0 1 0 0 0]

In [111]: output = pd.DataFrame({'PassengerId': test_data.PassengerId, 'Survived': predictions})
          output.to_csv('my_submission.csv', index=False)
          print("Your submission was successfully saved!")

          Your submission was successfully saved!
```

CONCLUSION

Our prediction had a Kaggle score of 0.798947 which is good. This prediction could be improved if we unlock more patterns to “Survived”.

You may realize we deleted row containing missing values in our train data but could not do same for that of the test data. Deleting values in our test data would render our prediction invalid because the prediction will be short of the rows deleted.