Titanic

By: Liz, Ericka, Kate, & Austin

Table Of Contents

Objective

- Create a machine learning model of the Titanic dataset
- See how factors such as socioeconomic status, sex, & age affect the survival rates of passengers

The Dataset

“The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.

The data has been split into two groups: training set (train.csv) test set (test.csv)

The training set should be used to build your machine learning models. For the training set, we provide the outcome (also known as the “ground truth”) for each passenger. Your model will be based on “features” like passengers’ gender and class. You can also use feature engineering to create new features.

The test set should be used to see how well your model performs on unseen data. For the test set, we do not provide the ground truth for each passenger. It is your job to predict these outcomes. For each passenger in the test set, use the model you trained to predict whether or not they survived the sinking of the Titanic.”

The Dataset

```
train=pd.read_csv('https://raw.githubusercontent.com/katieeehan20/Titanic_Project_Groupwork/main/train.csv')

train.head()
```

	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

```
test=pd.read_csv('https://raw.githubusercontent.com/katieeehan20/Titanic_Project_Groupwork/main/test.csv')
test.head()
```

	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

Data Cleaning

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [4]:
```python
def clean(train):
    train = train.drop(["Ticket", "Cabin", "Name", "PassengerId"], axis=1)
    cols = ["SibSp", "Parch", "Fare", "Age"]
    for col in cols:
        train[col].fillna(train[col].median(), inplace=True)
    train.Embarked.fillna("U", inplace=True)
    return train
train = clean(train)
test = clean(test)
```

In [5]:
```python
train.head()
```

Out[5]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S

In [6]:
```python
test.head()
```

Out[6]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	male	34.5	0	0	7.8292	Q
1	3	female	47.0	1	0	7.0000	S
2	2	male	62.0	0	0	9.6875	Q
3	3	male	27.0	0	0	8.6625	S
4	3	female	22.0	1	1	12.2875	S

To clean our data, we removed a few columns that didn't seem relevant to the survival score or were missing a significant amount of data. These values included the Ticket number, Cabin number, Name, and Passenger ID. Next, we filled in blank values with either the mean value of the column data or a character, "U", to represent an unknown value.

Data Cleaning Continued

For our string and character values, we had to convert them to numeric values in order to finish cleaning and analyze our data. Here we used SciKitLearn LabelEncoder() to assign a numeric index to each string or character value. We could have also used the map() or dict() functions. However, we went with the LabelEncoder() because it was more straightforward and automated.

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()

cols = ["Sex", "Embarked"]

for col in cols:
    train[col] = le.fit_transform(train[col])
    test[col] = le.transform(test[col])
    print(le.classes_)

train.head(5)
```

```
['female' 'male']
['C' 'Q' 'S' 'U']
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	22.0	1	0	7.2500	2
1	1	1	0	38.0	1	0	71.2833	0
2	1	3	0	26.0	0	0	7.9250	2
3	1	1	0	35.0	1	0	53.1000	2
4	0	3	1	35.0	0	0	8.0500	2

Analysis


```
only=train.loc[:,["Survived", "Pclass", "Age", "Sex"]]
only
```

	Survived	Pclass	Age	Sex
0	0	3	22.0	1
1	1	1	38.0	0
2	1	3	26.0	0
3	1	1	35.0	0
4	0	3	35.0	1
...	...	...	...	...
886	0	2	27.0	1
887	1	1	19.0	0
888	0	3	28.0	0
889	1	1	26.0	1
890	0	3	32.0	1

891 rows × 4 columns

```
onlysurvived=only[only["Survived"]==1]
onlysurvived
```

	Survived	Pclass	Age	Sex
1	1	1	38.0	0
2	1	3	26.0	0
3	1	1	35.0	0
8	1	3	27.0	0
9	1	2	14.0	0
...	...	...	...	...
875	1	3	15.0	0
879	1	1	56.0	0
880	1	2	25.0	0
887	1	1	19.0	0
889	1	1	26.0	1

342 rows × 4 columns

Analysis Continued

```
#Kate

only.describe()
```

	Survived	Pclass	Age	Sex
count	891.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.361582	0.647587
std	0.486592	0.836071	13.019697	0.477990
min	0.000000	1.000000	0.420000	0.000000
25%	0.000000	2.000000	22.000000	0.000000
50%	0.000000	3.000000	28.000000	1.000000
75%	1.000000	3.000000	35.000000	1.000000
max	1.000000	3.000000	80.000000	1.000000

```
# Kate
# this summarizes all the columns (numeric only)

only.describe(include='all')
```

	Survived	Pclass	Age	Sex
count	891.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.361582	0.647587
std	0.486592	0.836071	13.019697	0.477990
min	0.000000	1.000000	0.420000	0.000000
25%	0.000000	2.000000	22.000000	0.000000
50%	0.000000	3.000000	28.000000	1.000000
75%	1.000000	3.000000	35.000000	1.000000
max	1.000000	3.000000	80.000000	1.000000

Analysis Continued

```
#Kate

# 1- number of people survived

train["Survived"].value_counts().head()
```

```
0    549
1    342
Name: Survived, dtype: int64
```

```
#Kate

train["Sex"].value_counts().head()
```

```
1    577
0    314
Name: Sex, dtype: int64
```

Analysis Continued

Survival Rates Of Passengers

```
class1 = train.loc[train.Pclass == 1]["Survived"]
rate_class1 = round(sum(class1)/len(class1)*100,2)

print("Percent of first class who survived: %", rate_class1)
```

```
Percent of first class who survived: % 62.96
```

```
class2 = train.loc[train.Pclass == 2]["Survived"]
rate_class2 = round(sum(class2)/len(class2)*100,2)

print("Percent of second class who survived: %", rate_class2)
```

```
Percent of second class who survived: % 47.28
```

```
class3 = train.loc[train.Pclass == 3]["Survived"]
rate_class3 = round(sum(class3)/len(class3)*100,2)

print("Percent of third class who survived: %", rate_class3)
```

```
Percent of third class who survived: % 24.24
```

Data Visualization 01

```
#Kate

class1 = train.loc[train.Pclass == 1]["Survived"]
rate_class1 = round(sum(class1)/len(class1)*100,2)

print("Percent of first class who survived: %", rate_class1)
```

```
Percent of first class who survived: % 62.96
```

```
#Kate

class2 = train.loc[train.Pclass == 2]["Survived"]
rate_class2 = round(sum(class2)/len(class2)*100,2)

print("Percent of second class who survived: %", rate_class2
```

```
Percent of second class who survived: % 47.28
```

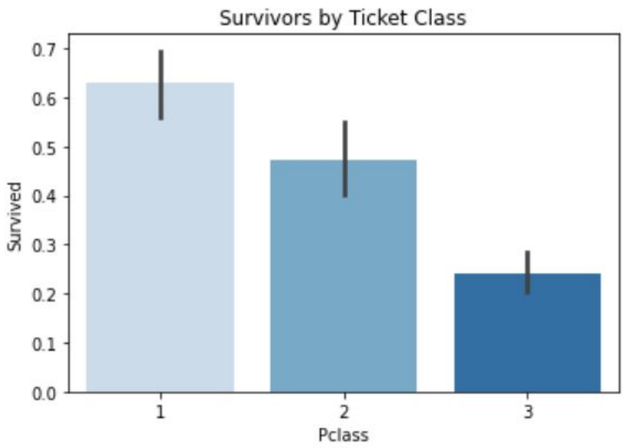
```
#Kate

class3 = train.loc[train.Pclass == 3]["Survived"]
rate_class3 = round(sum(class3)/len(class3)*100,2)

print("Percent of third class who survived: %", rate_class3)
```

```
Percent of third class who survived: % 24.24
```

```
sns.barplot( train["Pclass"], train["Survived"], palette= "Blues",data=train)
plt.title("Survivors by Ticket Class")
```

```
C:\Users\erick\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarn
gs: x, y. From version 0.12, the only valid positional argument will be `data`,
keyword will result in an error or misinterpretation.
  warnings.warn(
```

```
Text(0.5, 1.0, 'Survivors by Ticket Class')
```



Data Visualization 02

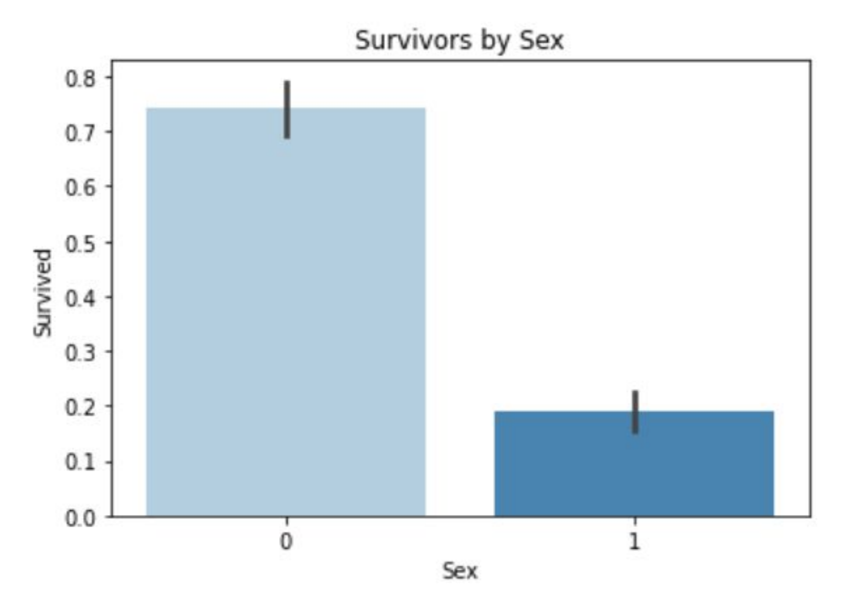
```
women = only.loc[only.Sex == 'female']["Survived"]

rate_women = sum(women)/len(women)

print("% of women who survived:", rate_women)
```

```
% of women who survived: 0.7420382165605095
```

```
men = only.loc[only.Sex == 'male']["Survived"]
rate_men = sum(men)/len(men)

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

```
% of men who survived: 0.18890814558058924
```

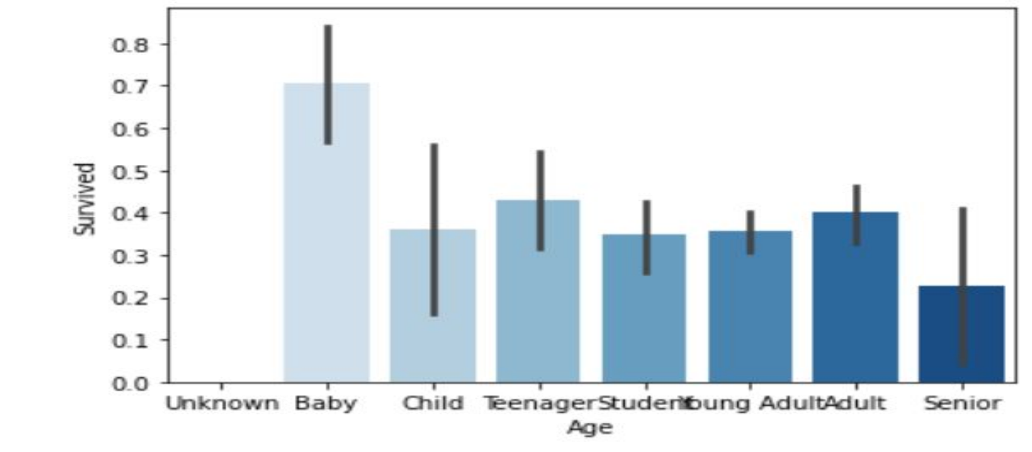
```
sns.barplot(only["Sex"], only["Survived"], data=train)
```



Data Visualization 03

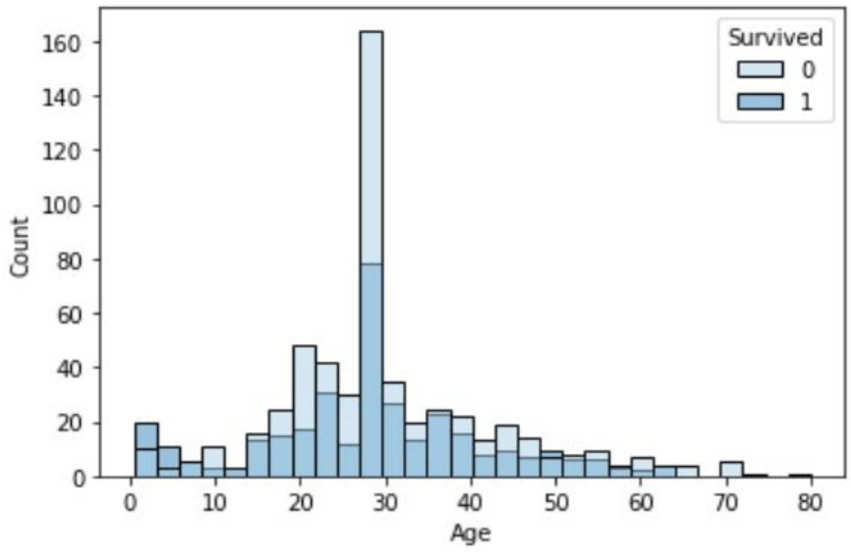
```
#Kate

only["Age"] = only["Age"].fillna(-0.5)
only["Age"] = only["Age"].fillna(-0.5)
bins = [-1, 0, 5, 12, 18, 24, 35, 60, np.inf]
labels = ['Unknown', 'Baby', 'Child', 'Teenager', 'Student', 'Young Adult', 'Adult', 'Senior']
only['Age'] = pd.cut(only["Age"], bins, labels = labels)
test['Age'] = pd.cut(test["Age"], bins, labels = labels)

sns.barplot(only["Age"], only["Survived"], palette = "Blues", data=train)
```

We grouped ages here categorically, into bins. Since we cleaned the data set in the beginning and filled all null values with the mean, we ended up with no unknown values.



Data Visualization 04

```
#Kate

sns.histplot(data=train, x='Age', hue="Survived", palette='Blues')
plt.show()
```

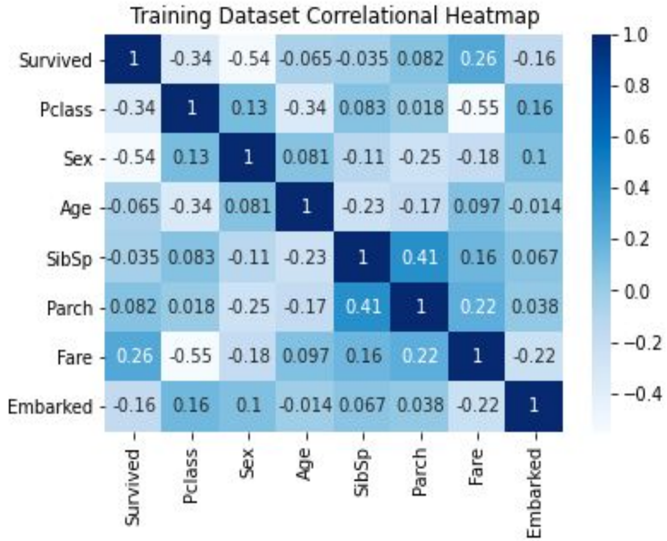


Histogram of people survived and died by age

Data Visualization 05

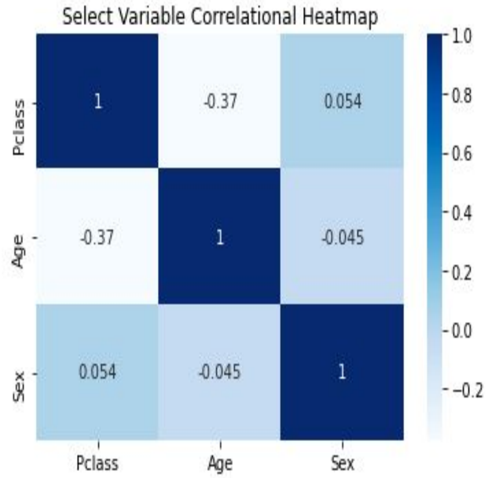
```python
corr = train.corr()
plt.title("Training Dataset Correlational Heatmap")
sns.heatmap(corr, cmap="Blues", annot=True)
```

<AxesSubplot:title={'center':'Training Dataset Correlational Heatmap'}>




```python
#correlation between all survivors age and class
corr = onlysurvived.corr()
#del onlysurvived["Survived"]
plt.title("Select Variable Correlational Heatmap")
sns.heatmap(corr, cmap="Blues", annot=True)
```

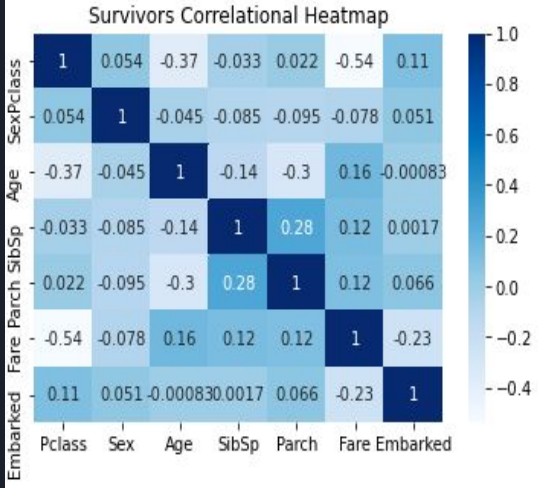
<AxesSubplot:title={'center':'Select Variable Correlational Heatmap'}>



Data Visualizations 06

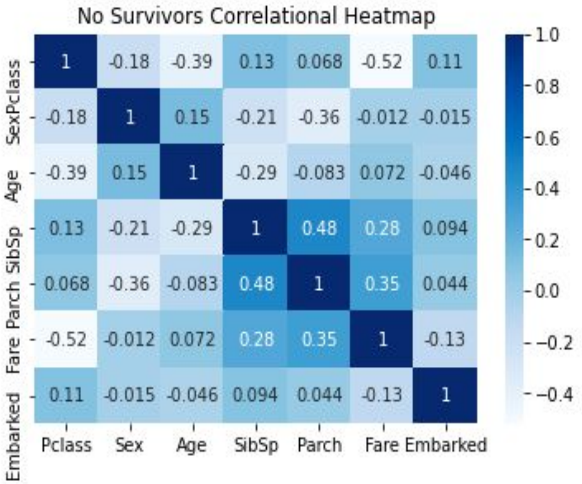
```
corr = allsurvivors.corr()
plt.title("Survivors Correlational Heatmap")
sns.heatmap(corr, cmap="Blues", annot=True)
```

```
<AxesSubplot:title={'center':'Survivors Correlational Heatmap'}>
```




```
corr = nosurvivors.corr()
plt.title("No Survivors Correlational Heatmap")
sns.heatmap(corr, cmap="Blues", annot=True)
```

```
<AxesSubplot:title={'center':'No Survivors Correlational Heatmap'}>
```



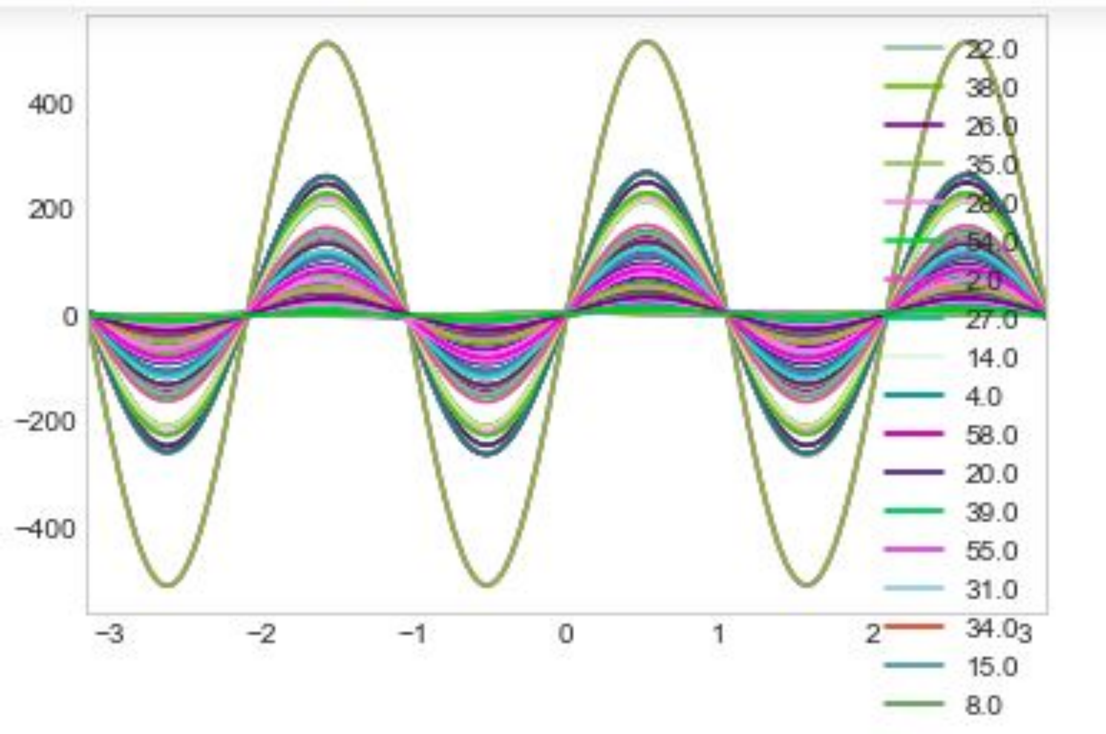
Data Visualizations 07

```
from pandas.plotting import parallel_coordinates

pd.plotting.andrews_curves(train, 'Age')
```

```
<AxesSubplot:>
```

Andrew Curves

➢ A dataset is represented in every curve
➢ Each color represents a different age
➢ Help display multidimensional data with many variables



Predictions

- Predict survival based on:
  - Pclass
  - Age
  - Sex
- 3 features for simplicity
- Utilized SciKit-Learn functions
  - Logistic Regression
  - Decision tree

	Survived	Pclass	Sex	Age
0	0	3	0	22
1	1	1	1	38
2	1	3	1	26
3	1	1	1	35
4	0	3	0	35
...	...	...	...	...
886	0	2	0	27
887	1	1	1	19
888	0	3	1	22
889	1	1	0	26
890	0	3	0	32

Prediction Problems and Solutions

- One main problem was the age group
- Null values made predictions difficult
- Solution: Fill null age values
  - Based on average age

Sex	Age
male	22.0
female	38.0
female	26.0
female	35.0
male	35.0
...	...
male	27.0
female	19.0
female	NaN
male	26.0
male	32.0

```python
#Adding random ages for null age

for dataset in train_test:
    #basic data collection
    average_age = dataset["Age"].mean()
    std_age = dataset["Age"].std()
    null_count = dataset["Age"].isnull().count()

    #calculation
    null_random_fill = np.random.randint(average_age - std_age, average_age + std_age, size=null_count)
    # Data fill in
    dataset["Age"][np.isnan(dataset["Age"])] = null_random_fill
    dataset["Age"] = dataset["Age"].astype(int)
```

Tested Models, Results, & Reflection

- Logistic Regression
- Decision Tree
- Logistic Regression avg: 79.01%
- Decision Tree avg: 87.79%

- Reflection: Incorporating a function for prediction averages

```
#Running Logistic Regression

eq = LogisticRegression()
eq.fit(x_training,y_training)
logistic_reg = eq.predict(x_testing)
final_log_reg = round( eq.score(x_training, y_training) * 100, 2)
print(str(final_log_reg) + " percent")
```

```
78.9 percent
```

```
#Running decision tree

eq_dt = DecisionTreeClassifier()
eq_dt.fit(x_training, y_training)
y_prediction_dt = eq_dt.predict(x_testing)
dt_out = round(eq_dt.score(x_training, y_training)* 100, 2)

dt_out
```

```
87.99
```

Conclusion

First class passengers had the highest chance of survival compared to second and third class passengers due to their social status, which definitely helped their accessibility. They were more likely to be informed by the crew about the situation and their cabins were closer to the lifeboats.

Women and children were also prioritized due to the captain's order. They were allowed to board the lifeboats.