```python
#Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
import seaborn as sns


train_data = pd.read_csv('/content/train_data_titanic.csv')

train_data.info()
train_data.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```


```
{"summary":"{\n  \"name\": \"train_data\",\n  \"rows\": 8,\n
\"fields\": [\n    {\n      \"column\": \"PassengerId\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
320.8159711429856,\n        \"min\": 1.0,\n        \"max\": 891.0,\n
\"num_unique_values\": 6,\n        \"samples\": [\n          891.0,\n
446.0,\n          668.5\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Survived\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 314.8713661874558,\n        \"min\":
0.0,\n        \"max\": 891.0,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          0.3838383838383838,\n          1.0,\n
0.4865924542648585\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Pclass\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 314.2523437079693,\n        \"min\": 0.8360712409770513,\n
\"max\": 891.0,\n        \"num_unique_values\": 6,\n
```

\"samples\": [\n   891.0,\n   2.308641975308642,\n
3.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Age\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 242.9056731818781,\n   \"min\": 0.42,\n   \"max\":
714.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n
29.69911764705882,\n   28.0,\n   714.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"SibSp\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 314.4908277465442,\n
\"min\": 0.0,\n   \"max\": 891.0,\n   \"num_unique_values\":
6,\n   \"samples\": [\n   891.0,\n
0.5230078563411896,\n   8.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Parch\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 314.65971717879,\n
\"min\": 0.0,\n   \"max\": 891.0,\n   \"num_unique_values\":
5,\n   \"samples\": [\n   0.38159371492704824,\n
6.0,\n   0.8060572211299559\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Fare\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 330.6256632228577,\n
\"min\": 0.0,\n   \"max\": 891.0,\n   \"num_unique_values\":
8,\n   \"samples\": [\n   32.204207968574636,\n
14.4542,\n   891.0\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe"}

DATA PREPARATION

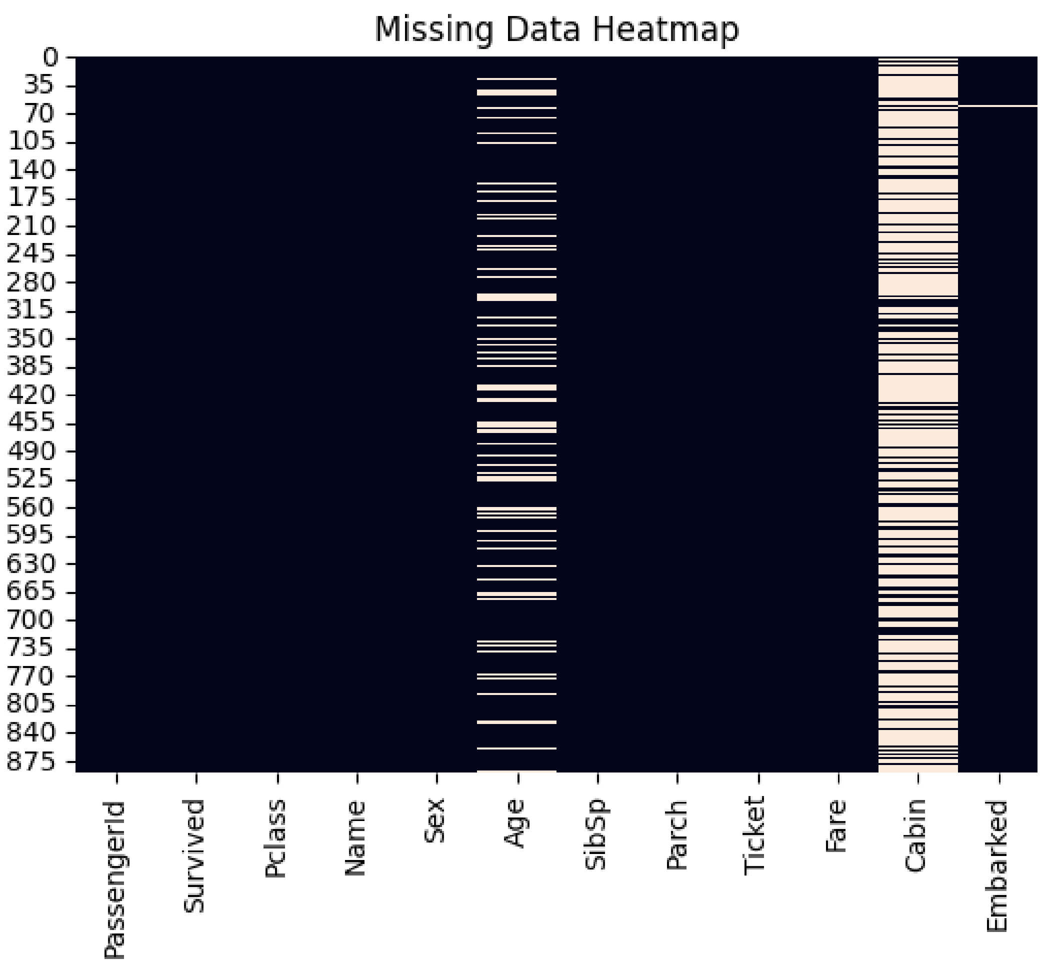
When using linear regression as a predictive model, it is extremely important to remember that the ranges for all attributes in the scoring data must be within the ranges for the corresponding attributes in the training data. This is because a training data set cannot be relied upon to predict a target attrtibute for observations whose values fall outside the training data set's values.

```
# Check for missing values
print("Missing values in training data:\n", train_data.isnull().sum())
```

```
Missing values in training data:
 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
```

```
sns.heatmap(train_data.isnull(), cbar=False)
plt.title('Missing Data Heatmap')
plt.show()
```



HANDLING MISSING VALUES

```
train_data['Age'] = train_data['Age'].fillna(train_data['Age'].mean())
train_data = train_data.dropna(subset=['Embarked'])

# Check for missing values
print("Missing values in training data:\n", train_data.isnull().sum())
```

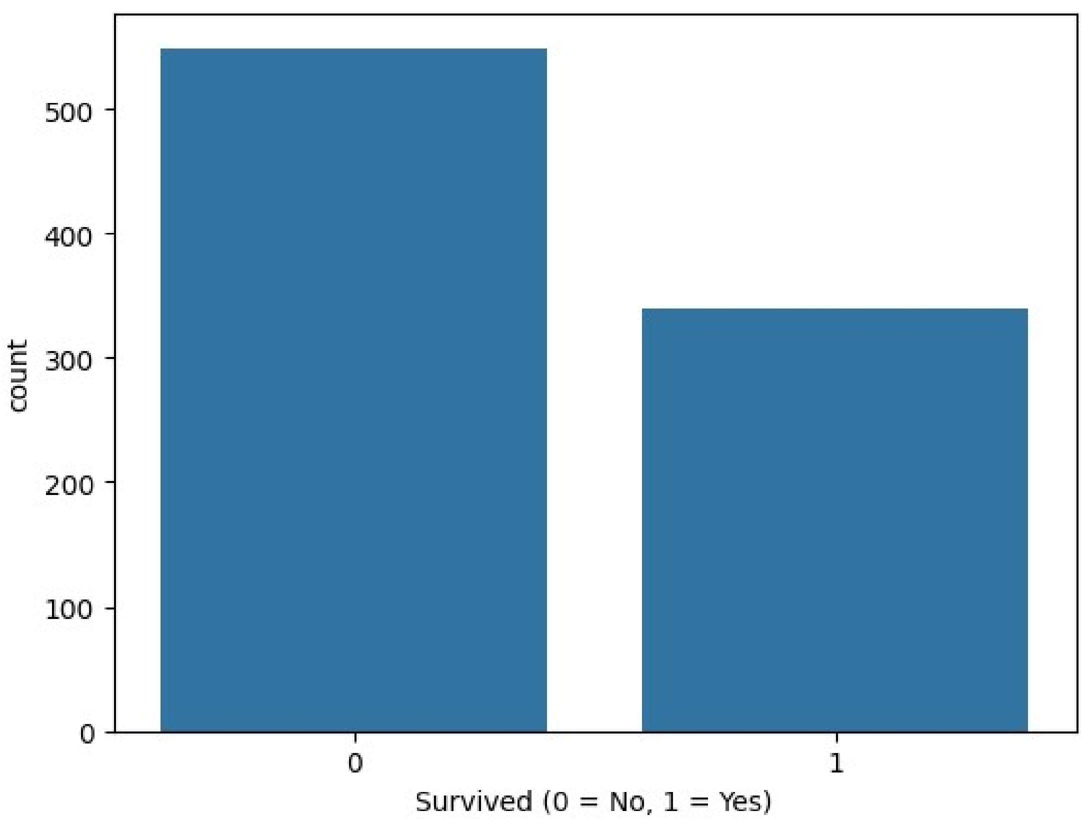
```
Missing values in training data:
 PassengerId      0
Survived         0
```

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

The Missing Values in the Cabin were not removed as they represent more than 50% of the data set. Removing this would potentially affect the accuracy the result.

Visualize key features like Age, Fare, and Survived

```
sns.countplot(x='Survived', data=train_data)
plt.xlabel("Survived (0 = No, 1 = Yes)")

Text(0.5, 0, 'Survived (0 = No, 1 = Yes)')
```



```
train_data = pd.get_dummies(train_data, columns=['Sex',
'Embarked'],drop_first=True)
display(train_data)
```

{"summary":"{\n \"name\": \"train_data\",\n \"rows\": 889,\n \"fields\": [\n   {\n   \"column\": \"PassengerId\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 256,\n   \"min\": 1,\n   \"max\": 891,\n   \"num_unique_values\": 889,\n   \"samples\": [\n   282,\n   436,\n   40\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Survived\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   1,\n   0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Pclass\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 1,\n   \"max\": 3,\n   \"num_unique_values\": 3,\n   \"samples\": [\n   3,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Name\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 889,\n   \"samples\": [\n   \"Olsson, Mr. Nils Johan Goransson\",\n   \"Carter, Miss. Lucile Polk\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 12.968366309252332,\n   \"min\": 0.42,\n   \"max\": 80.0,\n   \"num_unique_values\": 89,\n   \"samples\": [\n   59.0,\n   36.5\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"SibSp\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1,\n   \"min\": 0,\n   \"max\": 8,\n   \"num_unique_values\": 7,\n   \"samples\": [\n   1,\n   0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Parch\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 6,\n   \"num_unique_values\": 7,\n   \"samples\": [\n   0,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Ticket\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 680,\n   \"samples\": [\n   \"11774\",\n   \"29105\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Fare\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 49.69750431670801,\n   \"min\": 0.0,\n   \"max\": 512.3292,\n   \"num_unique_values\": 247,\n   \"samples\": [\n   11.2417,\n   51.8625\n   ],

```
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Cabin\",\n      \"properties\": {\
n        \"dtype\": \"category\",\n        \"num_unique_values\":
146,\n        \"samples\": [\n          \"C125\",\n          \"C101\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Sex_male\",\n      \"properties\": {\n        \"dtype\":
\"boolean\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          false,\n          true\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Embarked_Q\",\n
\"properties\": {\n        \"dtype\": \"boolean\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          true,\n
false\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Embarked_S\",\n      \"properties\": {\n        \"dtype\":
\"boolean\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          false,\n          true\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"train_data"}
```


```
X = train_data.drop(columns=['Survived', 'Name', 'Ticket', 'Cabin'])
y = train_data['Survived']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

MODELING

Logistic regression is an excellent way to predict whether or not something will happen, and how confident we are in such predictions. It takes a number of numeric attributes into account and then uses those through a training data set to predict the probable outcomes in a comparable scoring data set. Logistic regression uses a nominal target attribute to categorize observations in a scoring data set into their probable outcomes.

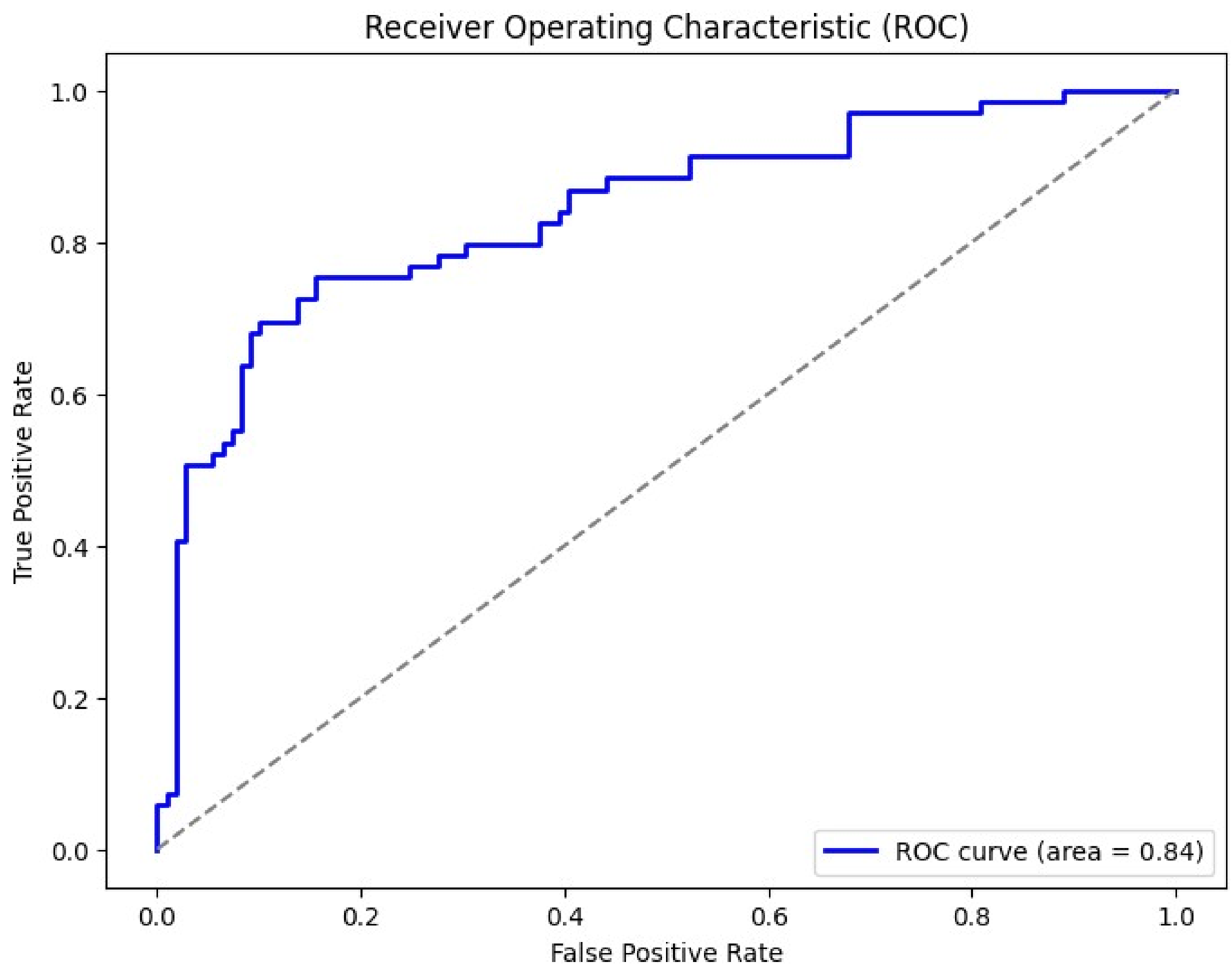
As with linear regression, the scoring data must have ranges that fall within their corresponding training data ranges.

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:")
```

```python
print(confusion_matrix(y_test, y_pred))
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.797752808988764
Confusion Matrix:
[[92 17]
 [19 50]]
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.84      0.84       109
           1       0.75      0.72      0.74        69

    accuracy                           0.80       178
   macro avg       0.79      0.78      0.79       178
weighted avg       0.80      0.80      0.80       178


/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/
_logistic.py:469: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```python
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, model.predict_proba(X_test)
[:,1])
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (area =
{roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc='lower right')
plt.show()
```




```
score_data = pd.read_csv('/content/score_data_titanic.csv')

score_data.info()
score_data.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  4 non-null      int64
 1   Pclass       4 non-null      int64
 2   Name         4 non-null      object
 3   Sex          4 non-null      object
 4   Age          4 non-null      int64
 5   SibSp        4 non-null      int64
 6   Parch        4 non-null      int64
 7   Fare         4 non-null      float64
 8   Embarked     4 non-null      object
```

```
dtypes: float64(1), int64(5), object(3)
memory usage: 416.0+ bytes
```


{"summary":"{\n \"name\": \"score_data\",\n \"rows\": 8,\n
\"fields\": [\n   {\n   \"column\": \"PassengerId\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
412.3871121772022,\n   \"min\": 1.2909944487358056,\n
\"max\": 895.0,\n   \"num_unique_values\": 7,\n
\"samples\": [\n   4.0,\n   893.5,\n   894.25\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Pclass\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\":
1.0492915798561668,\n   \"min\": 0.9574271077563381,\n
\"max\": 4.0,\n   \"num_unique_values\": 7,\n   \"samples\":
[\n   4.0,\n   2.25,\n   2.5\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 18.84899347844133,\n
\"min\": 4.0,\n   \"max\": 62.0,\n   \"num_unique_values\":
8,\n   \"samples\": [\n   43.25,\n   40.5,\n
4.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"SibSp\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 1.3562026818605375,\n   \"min\": 0.0,\n   \"max\":
4.0,\n   \"num_unique_values\": 5,\n   \"samples\": [\n
0.25,\n   1.0,\n   0.5\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Parch\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 1.4142135623730951,\
n   \"min\": 0.0,\n   \"max\": 4.0,\n
\"num_unique_values\": 2,\n   \"samples\": [\n   0.0,\n
4.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Fare\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 3.046554516851229,\n   \"min\": 1.3070322617798436,\n
\"max\": 10.5,\n   \"num_unique_values\": 7,\n
\"samples\": [\n   4.0,\n   8.75\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe"}

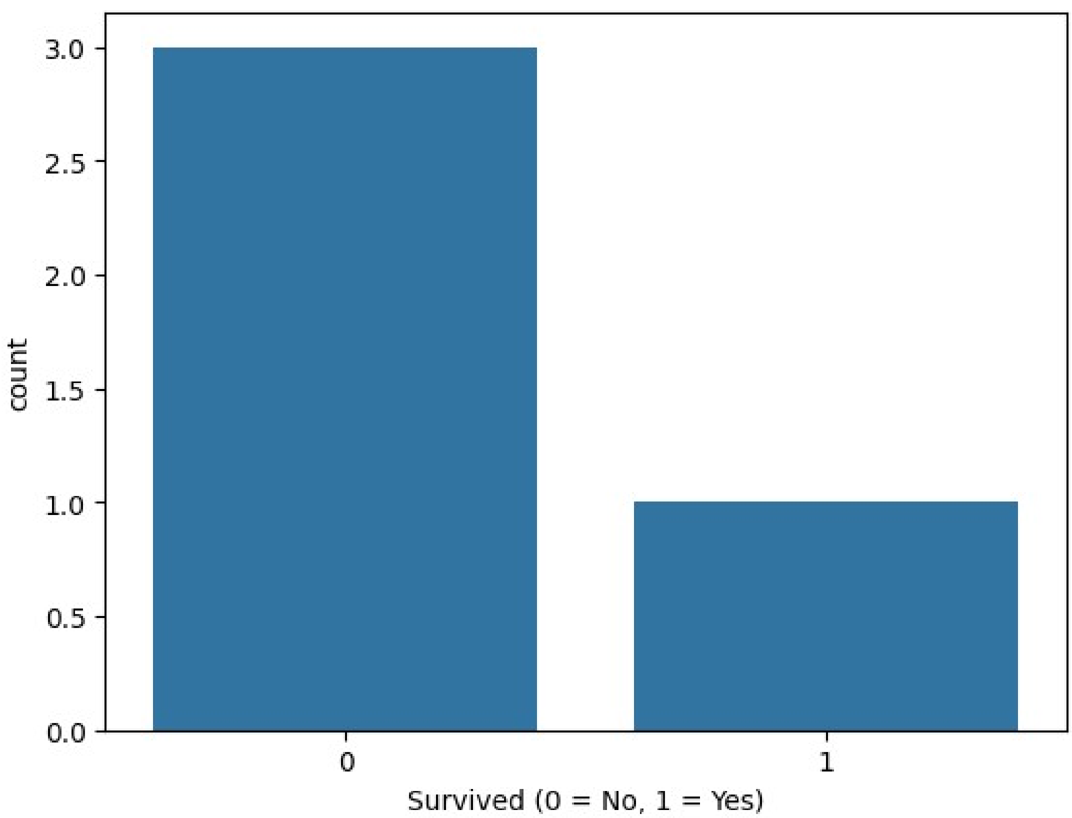

```python
score_data = score_data.drop(columns=['PassengerId'])

score_data = pd.get_dummies(score_data, columns=['Sex', 'Embarked'],
drop_first=True)

score_data['Age'] = score_data['Age'].fillna(score_data['Age'].mean())
score_data['Fare'] =
score_data['Fare'].fillna(score_data['Fare'].mean())
score_data = score_data.reindex(columns=X.columns, fill_value=0)
predictions = model.predict(score_data)
```

```
score_data['PassengerId'] =
pd.read_csv('/content/score_data_titanic.csv')['PassengerId']


score_data['Survived'] = predictions
score_data[['PassengerId', 'Survived']].to_csv('predictions.csv',
index=False)

# Display the predictions with PassengerId
print(score_data[['PassengerId', 'Survived']])
```

```
   PassengerId  Survived
0          892         0
1          893         0
2          894         0
3          895         1
```

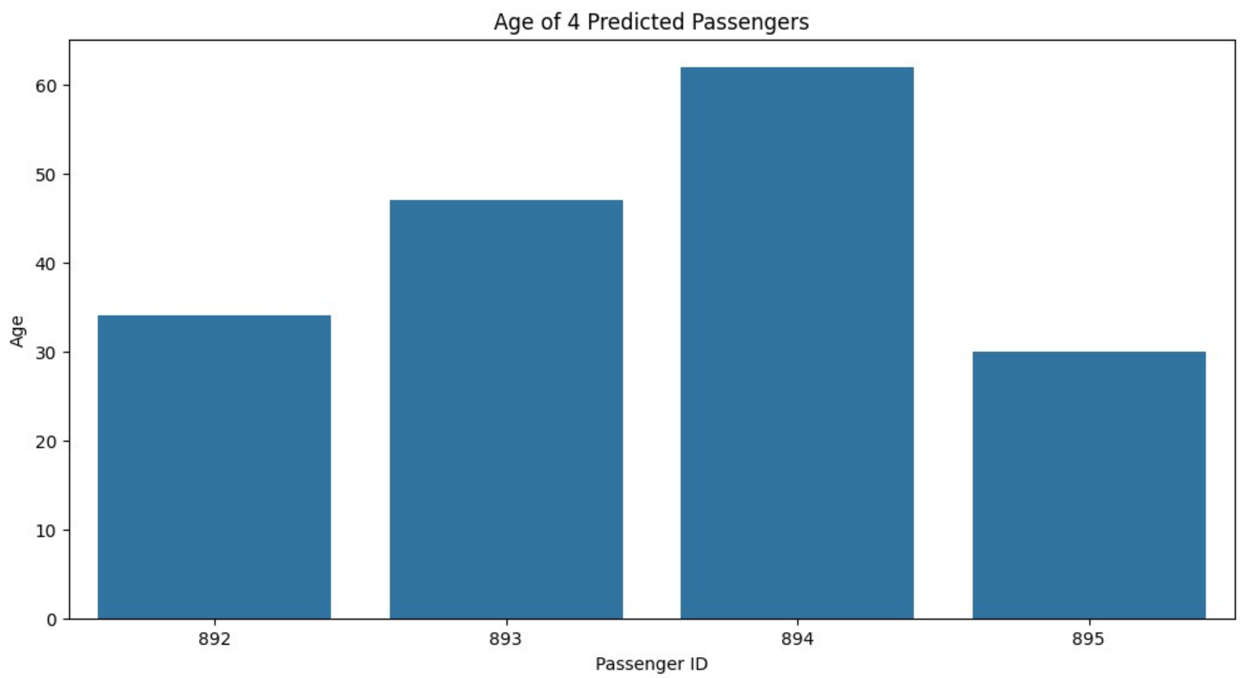
```
sns.countplot(x='Survived', data=score_data)
plt.xlabel("Survived (0 = No, 1 = Yes)")
```

```
Text(0.5, 0, 'Survived (0 = No, 1 = Yes)')
```

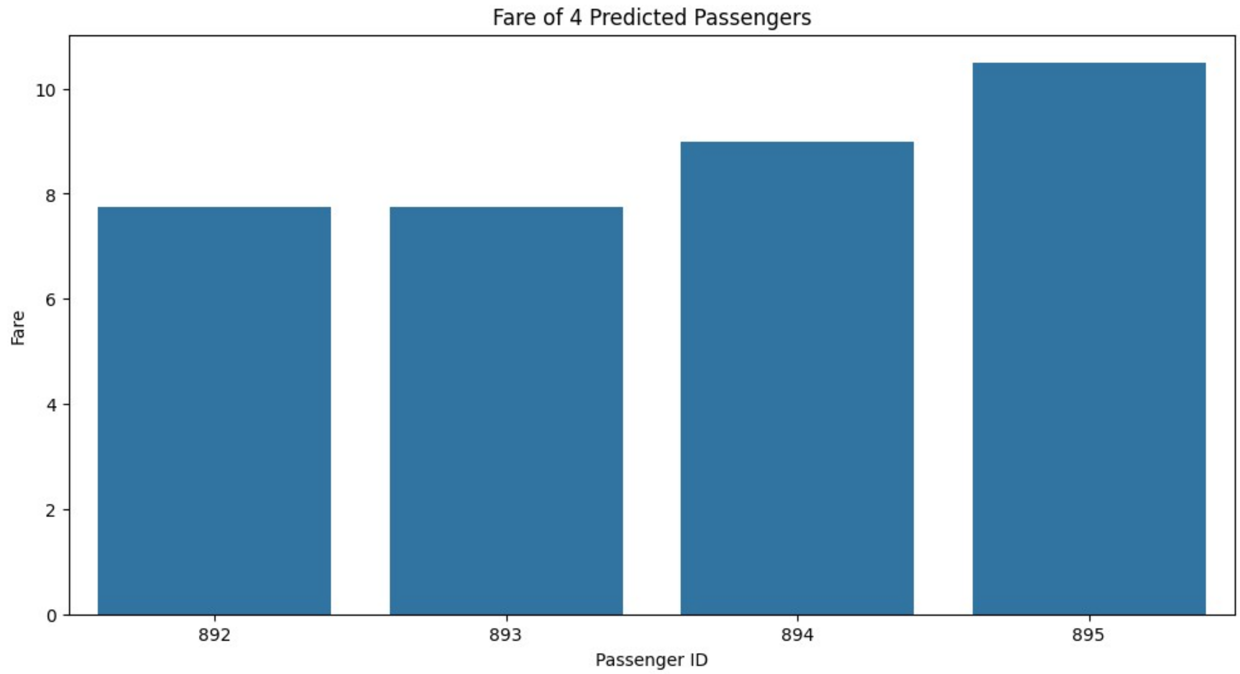



```
# Select the relevant columns for visualization
predicted_passengers = score_data[['PassengerId', 'Sex_male', 'Age',
```

```python
'Fare', 'Embarked_S', 'Survived']]

# Set the figure size for a better view
plt.figure(figsize=(12, 6))

# Plot the data for these passengers
sns.barplot(x='PassengerId', y='Age', data=predicted_passengers)
plt.title('Age of 4 Predicted Passengers')
plt.xlabel('Passenger ID')
plt.ylabel('Age')
plt.show()
```




```python
# Barplot for Fare of 4 predicted passengers
plt.figure(figsize=(12, 6))
sns.barplot(x='PassengerId', y='Fare', data=predicted_passengers)
plt.title('Fare of 4 Predicted Passengers')
plt.xlabel('Passenger ID')
plt.ylabel('Fare')
plt.show()
```



CONCLUSION

Based in the predicted model, there were 3 passengers who would not survive, while there was only 1 passenger who is likely to survive the titanic.

FINDINGS

Based on the visual presentation of the predicted passengers' Age, Fare, and Gender, Survival Status is likely influenced by the mentioned factors. Passengers who are much younger and have paid higher fares have higher chance of survival, While those who are much older and have paid lower fares have the lower chance of survival.