Titanic Survial Predicition in python (Machine Learning project)

Required Libraries

In [7]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Step 1: Data frames Loading and importing

In [5]:
```
df=pd.read_csv("C:/Users/DELL/Desktop/A/day 6(project)/train.csv")
```

In [6]:
```
titanic_data = pd.read_csv("C:/Users/DELL/Desktop/A/day 6(project)/train.csv")
```

In [16…
```
titanic_data.head()
```

Out[16…

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	male	22.0	1	0	7.2500	S
1	2	1	1	female	38.0	1	0	71.2833	C
2	3	1	3	female	26.0	0	0	7.9250	S
3	4	1	1	female	35.0	1	0	53.1000	S
4	5	0	3	male	35.0	0	0	8.0500	S

In [9]:
```
titanic_data.info()
```

```
<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
```

In [10…

```
titanic_data.describe()
```

Out[10…

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

shape of data sheet

In [11…
```
titanic_data.shape
```

Out[11…
```
(891, 12)
```

In [12…
```
titanic_data.columns
```

Out[12…
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

Checking for Missing values

Step 2: Data Cleaning

In [13…
```
titanic_data.isnull().sum()
```

Out[13…

```
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
```

In [14…

```
print(titanic_data.columns)
```

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

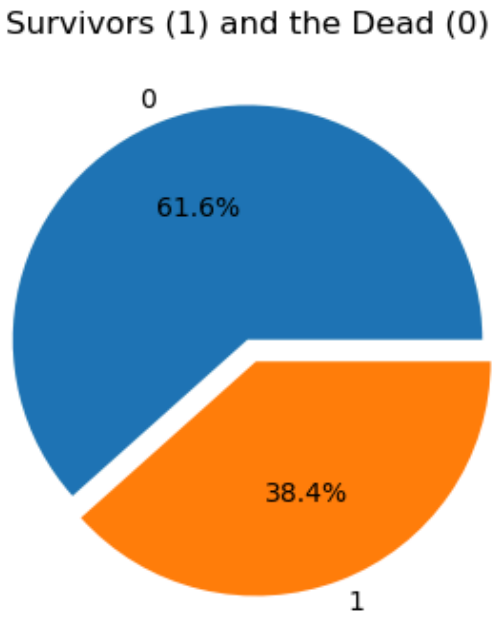
In [17…

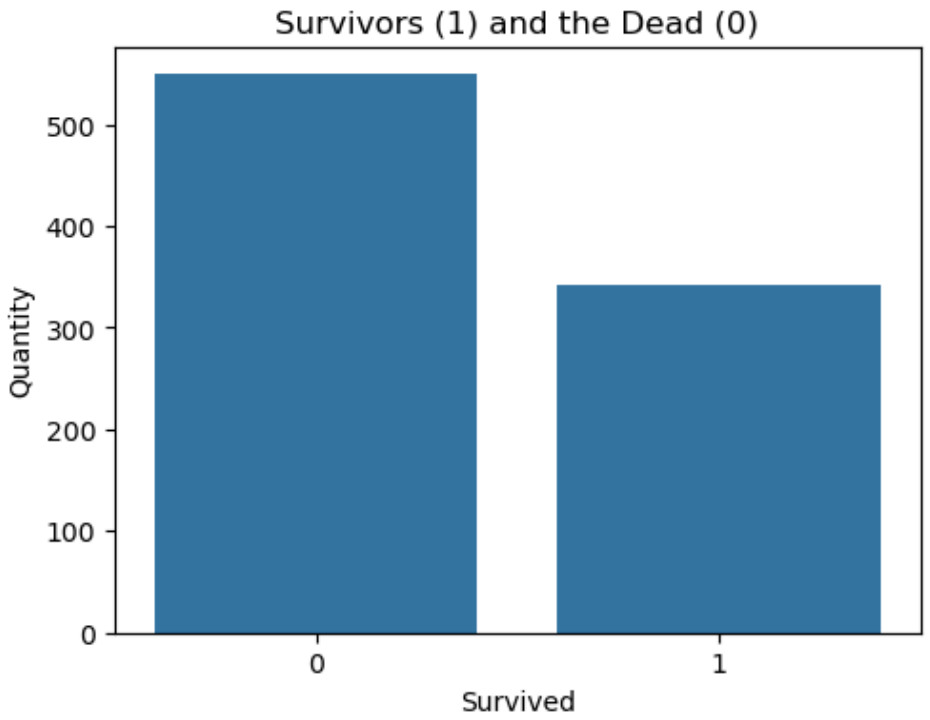
```
titanic_data = pd.read_csv("C:/Users/DELL/Desktop/A/day 6(project)/train.csv")

# Now safely drop columns
titanic_data.drop(['Ticket', 'Cabin', 'Name'], axis=1, inplace=True, errors='ignore')
```

In [21…

```
import matplotlib.pyplot as plt
import seaborn as sns

f, ax = plt.subplots(1, 2, figsize=(12, 4))

titanic_data['Survived'].value_counts().plot.pie(
    explode=[0, 0.1], autopct='%1.1f%%', ax=ax[0], shadow=False)

ax[0].set_title('Survivors (1) and the Dead (0)')
ax[0].set_ylabel('')

sns.countplot(x='Survived', data=titanic_data, ax=ax[1])
```

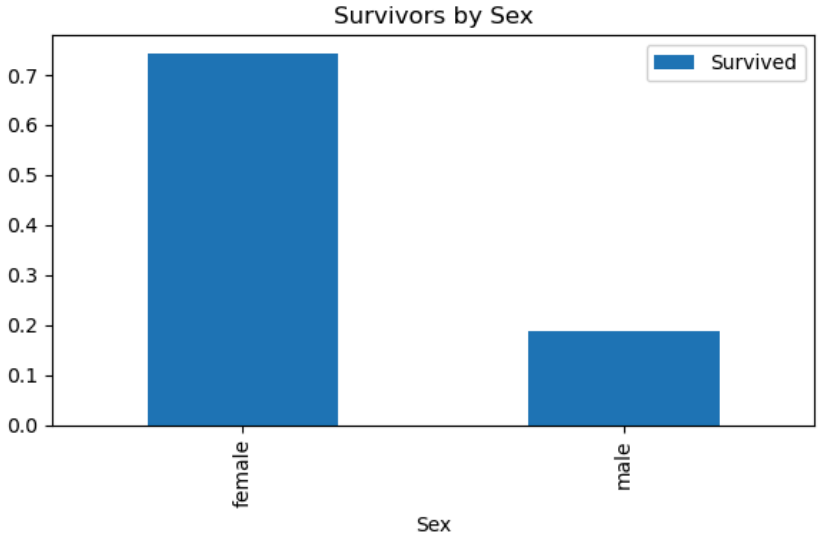
```
ax[1].set_ylabel('Quantity')
ax[1].set_title('Survivors (1) and the Dead (0)')
plt.show()
```

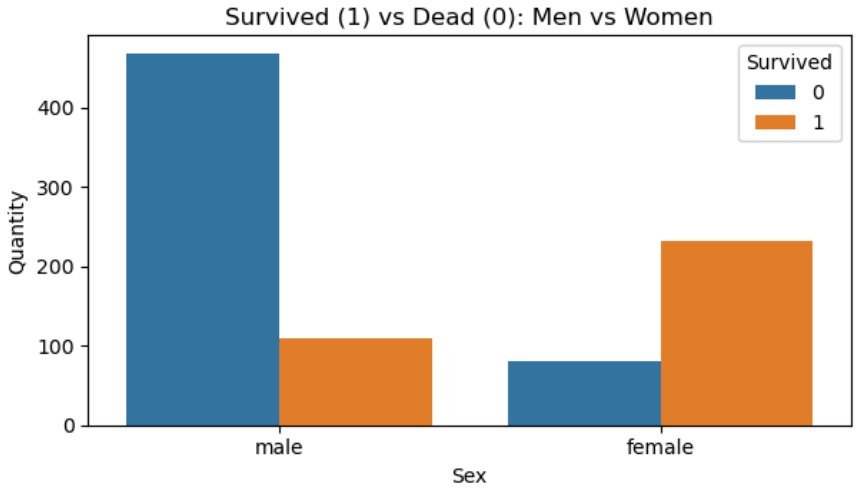




In [23…

```
f, ax = plt.subplots(1, 2, figsize=(12, 4))

# Plot mean survival rate by gender
titanic_data[['Sex', 'Survived']].groupby(['Sex']).mean().plot.bar(ax=ax[0])
ax[0].set_title('Survivors by Sex')

# Countplot with hue for Survived
sns.countplot(x='Sex', hue='Survived', data=titanic_data, ax=ax[1])
ax[1].set_ylabel('Quantity')
ax[1].set_title('Survived (1) vs Dead (0): Men vs Women')
plt.tight_layout()
plt.show()
```





Step 3 : Feature Engineering (Optimizing Data for Model Training) next step

In [9]:
```
train = pd.read_csv(r"C:\Users\DELL\Desktop\A\day 6(project)\train.csv")
test = pd.read_csv(r"C:\Users\DELL\Desktop\A\day 6(project)\test.csv")
```

In [ ]:
```
# Drop unnecessary columns
train = train.drop(['Ticket', 'Name'], axis=1)
test = test.drop(['Ticket', 'Name'], axis=1)
```

In [8]:
```
train = pd.read_csv("C:/Users/DELL/Desktop/A/day 6(project)/train.csv")
test = pd.read_csv("C:/Users/DELL/Desktop/A/day 6(project)/test.csv")
```

In [10…
```
train = train.drop(['Ticket', 'Name'], axis=1)
test = test.drop(['Ticket', 'Name'], axis=1)
```

In [11…
```
test
```

Out[11…

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	892	3	male	34.5	0	0	7.8292	NaN	Q
1	893	3	female	47.0	1	0	7.0000	NaN	S
2	894	2	male	62.0	0	0	9.6875	NaN	Q
3	895	3	male	27.0	0	0	8.6625	NaN	S
4	896	3	female	22.0	1	1	12.2875	NaN	S
...	...	...	...	...	...	...	...	...	...
413	1305	3	male	NaN	0	0	8.0500	NaN	S
414	1306	1	female	39.0	0	0	108.9000	C105	C
415	1307	3	male	38.5	0	0	7.2500	NaN	S
416	1308	3	male	NaN	0	0	8.0500	NaN	S
417	1309	3	male	NaN	1	1	22.3583	NaN	C

418 rows × 9 columns

In [12…

```
train
```

Out[12…

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	1	0	3	male	22.0	1	0	7.2500	NaN	S
1	2	1	1	female	38.0	1	0	71.2833	C85	C
2	3	1	3	female	26.0	0	0	7.9250	NaN	S
3	4	1	1	female	35.0	1	0	53.1000	C123	S
4	5	0	3	male	35.0	0	0	8.0500	NaN	S
…	…	…	…	…	…	…	…	…	…	…
886	887	0	2	male	27.0	0	0	13.0000	NaN	S
887	888	1	1	female	19.0	0	0	30.0000	B42	S
888	889	0	3	female	NaN	1	2	23.4500	NaN	S
889	890	1	1	male	26.0	0	0	30.0000	C148	C
890	891	0	3	male	32.0	0	0	7.7500	NaN	Q

891 rows × 10 columns

In [13…

```
train.isnull().sum()
```

Out[13…
```
PassengerId      0
Survived         0
Pclass           0
Sex              0
Age            177
SibSp            0
Parch            0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [14…
```
test.isnull().sum()
```

Out[14…
```
PassengerId      0
Pclass           0
Sex              0
Age             86
SibSp            0
Parch            0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

In [16…
```
train['Age'] = train['Age'].fillna(train['Age'].median())
test['Age'] = test['Age'].fillna(test['Age'].median())

train['Embarked'] = train['Embarked'].fillna(train['Embarked'].mode()[0])
test['Fare'] = test['Fare'].fillna(test['Fare'].median())
```

In [20…
```
train['Embarked'] = train['Embarked'].fillna(train['Embarked'].mode()[0])
```

In [21…
```
test
```

Out[21…

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	892	3	male	34.5	0	0	7.8292	NaN	Q
1	893	3	female	47.0	1	0	7.0000	NaN	S
2	894	2	male	62.0	0	0	9.6875	NaN	Q
3	895	3	male	27.0	0	0	8.6625	NaN	S
4	896	3	female	22.0	1	1	12.2875	NaN	S
…	…	…	…	…	…	…	…	…	…
413	1305	3	male	27.0	0	0	8.0500	NaN	S
414	1306	1	female	39.0	0	0	108.9000	C105	C
415	1307	3	male	38.5	0	0	7.2500	NaN	S
416	1308	3	male	27.0	0	0	8.0500	NaN	S
417	1309	3	male	27.0	1	1	22.3583	NaN	C

418 rows × 9 columns

In [22…

```
train
```

Out[22...

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	1	0	3	male	22.0	1	0	7.2500	NaN	S
1	2	1	1	female	38.0	1	0	71.2833	C85	C
2	3	1	3	female	26.0	0	0	7.9250	NaN	S
3	4	1	1	female	35.0	1	0	53.1000	C123	S
4	5	0	3	male	35.0	0	0	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	male	27.0	0	0	13.0000	NaN	S
887	888	1	1	female	19.0	0	0	30.0000	B42	S
888	889	0	3	female	28.0	1	2	23.4500	NaN	S
889	890	1	1	male	26.0	0	0	30.0000	C148	C
890	891	0	3	male	32.0	0	0	7.7500	NaN	Q

891 rows × 10 columns

In [23...

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

In [25...

```
print("Train columns:", train.columns)
print("Test columns:", test.columns)
```

```
Train columns: Index(['PassengerId', 'Survived', 'Pclass', 'Age', 'SibSp', 'Parch', 'Fare',
       'Cabin', 'Sex_male', 'Embarked_Q', 'Embarked_S'],
      dtype='object')
Test columns: Index(['PassengerId', 'Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'Cabin',
       'Sex_male', 'Embarked_Q', 'Embarked_S'],
      dtype='object')
```

In [26…
```
combined = pd.concat([train, test], sort=False)
```

In [28…
```
combined = pd.concat([train, test], sort=False)
```

In [29…
```
train = combined.loc[train.index]
test = combined.loc[test.index]
```

In [30…
```
test
```

Out[30…

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	Cabin	Sex_male	Embarked_Q	Embarked_S
0	1	0.0	3	22.0	1	0	7.2500	NaN	True	False	True
0	892	NaN	3	34.5	0	0	7.8292	NaN	True	True	False
1	2	1.0	1	38.0	1	0	71.2833	C85	False	False	False
1	893	NaN	3	47.0	1	0	7.0000	NaN	False	False	True
2	3	1.0	3	26.0	0	0	7.9250	NaN	False	False	True
…	…	…	…	…	…	…	…	…	…	…	…
415	1307	NaN	3	38.5	0	0	7.2500	NaN	True	False	True
416	417	1.0	2	34.0	1	1	32.5000	NaN	False	False	True
416	1308	NaN	3	27.0	0	0	8.0500	NaN	True	False	True
417	418	1.0	2	18.0	0	2	13.0000	NaN	False	False	True
417	1309	NaN	3	27.0	1	1	22.3583	NaN	True	False	False

836 rows × 11 columns

In [31…
```
train
```

Out[31…

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	Cabin	Sex_male	Embarked_Q	Embarked_S
0	1	0.0	3	22.0	1	0	7.2500	NaN	True	False	True
0	892	NaN	3	34.5	0	0	7.8292	NaN	True	True	False
1	2	1.0	1	38.0	1	0	71.2833	C85	False	False	False
1	893	NaN	3	47.0	1	0	7.0000	NaN	False	False	True
2	3	1.0	3	26.0	0	0	7.9250	NaN	False	False	True
…	…	…	…	…	…	…	…	…	…	…	…
886	887	0.0	2	27.0	0	0	13.0000	NaN	True	False	True
887	888	1.0	1	19.0	0	0	30.0000	B42	False	False	True
888	889	0.0	3	28.0	1	2	23.4500	NaN	False	False	True
889	890	1.0	1	26.0	0	0	30.0000	C148	True	False	False
890	891	0.0	3	32.0	0	0	7.7500	NaN	True	True	False

1309 rows × 11 columns

In [32…

```
X_train = train.drop(['PassengerId', 'Survived', 'Cabin'], axis=1)
y_train = train['Survived']

X_test = test.drop(['PassengerId', 'Cabin'], axis=1)
```

```
from sklearn.linear_model import LogisticRegression from sklearn.metrics import accuracy_score model =
LogisticRegression(max_iter=1000) model.fit(X_train, y_train) # Predict on train to check accuracy train_preds =
model.predict(X_train) print("Training Accuracy:", accuracy_score(y_train, train_preds))
```

In [34…

```
print(train['Survived'].isnull().sum())
```

418

In [37…
```
print(" train",train.head())  # Check what data you have in 'train'
print("test",test.head())   # Check test data
```

```
 train    PassengerId  Survived  Pclass   Age  SibSp  Parch     Fare Cabin  Sex_male  \
0            1       0.0       3  22.0      1      0   7.2500   NaN      True
0          892       NaN       3  34.5      0      0   7.8292   NaN      True
1            2       1.0       1  38.0      1      0  71.2833   C85     False
1          893       NaN       3  47.0      1      0   7.0000   NaN     False
2            3       1.0       3  26.0      0      0   7.9250   NaN     False

   Embarked_Q  Embarked_S
0       False        True
0        True       False
1       False       False
1       False        True
2       False        True
test    PassengerId  Survived  Pclass   Age  SibSp  Parch     Fare Cabin  Sex_male  \
0            1       0.0       3  22.0      1      0   7.2500   NaN      True
0          892       NaN       3  34.5      0      0   7.8292   NaN      True
1            2       1.0       1  38.0      1      0  71.2833   C85     False
1          893       NaN       3  47.0      1      0   7.0000   NaN     False
2            3       1.0       3  26.0      0      0   7.9250   NaN     False

   Embarked_Q  Embarked_S
0       False        True
0        True       False
1       False       False
1       False        True
2       False        True
```

In [38…
```
train = pd.read_csv(r"C:\Users\DELL\Desktop\A\day 6(project)\train.csv")
test = pd.read_csv(r"C:\Users\DELL\Desktop\A\day 6(project)\test.csv")
```

In [44…
```
print("Train Survived nulls:", train['Survived'].isnull().sum())  # should be 0
print("Test columns:", test.columns)  # Should NOT have Survived column
```

```
Train Survived nulls: 0
Test columns: Index(['PassengerId', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch',
       'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [45…
```python
combined = pd.concat([train.drop(columns=['Survived']), test], ignore_index=True)
# Do your feature engineering on combined here, then split later.
```

In [46…
```python
train_processed = combined.iloc[:len(train), :]
test_processed = combined.iloc[len(train):, :]
y_train = train['Survived']  # Keep target separate
```

In [47…
```python
X_train = train_processed
y_train = train['Survived']
X_test = test_processed
```

In [48…
```python
print(X_train.isnull().sum().sum())  # should be 0
print(X_test.isnull().sum().sum())   # should be 0
print(y_train.isnull().sum())        # should be 0
```

```
866
414
0
```

In [49…
```python
print("X_train missing values:")
print(X_train.isnull().sum()[X_train.isnull().sum() > 0])

print("\nX_test missing values:")
print(X_test.isnull().sum()[X_test.isnull().sum() > 0])
```

```
X_train missing values:
Age         177
Cabin       687
Embarked      2
dtype: int64

X_test missing values:
Age        86
Fare        1
Cabin     327
dtype: int64
```

In [56…
```python
X_train.drop(columns=['Cabin'], inplace=True, errors='ignore')
X_test.drop(columns=['Cabin'], inplace=True, errors='ignore')
```

In [54…
```python
pd.options.mode.chained_assignment = None  # 👈 disables the warning
```

In [57…
```python
print("Cabin" in X_train.columns)  # False = already gone
```

```
False
```

In [59…
```python
# Drop columns not useful for model and not numerical
X_train = X_train.select_dtypes(include=['number', 'bool'])
X_test = X_test.select_dtypes(include=['number', 'bool'])
```

Model Tranining

In [61…
```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

train_preds = model.predict(X_train)
print("Train Accuracy:", accuracy_score(y_train, train_preds))
```

```
Train Accuracy: 0.7025813692480359
```

In [60…
```
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

Out[60…
```
LogisticRegression
LogisticRegression(max_iter=1000)
```

In [63…
```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, X_train, y_train, cv=5)
print("Cross-validated accuracy:", scores.mean())
```

```
Cross-validated accuracy: 0.6891720544849664
```

In [64…
```
test_preds = model.predict(X_test)
submission = pd.DataFrame({
    'PassengerId': test['PassengerId'],
    'Survived': test_preds
})
submission.to_csv('submission.csv', index=False)
```

In [71…
```
import os

folder_path = r"D:\ML Projects\Titanic2"
os.makedirs(folder_path, exist_ok=True)
submission_path = os.path.join(folder_path, "submission.csv")
submission.to_csv(submission_path, index=False)

print(f"Submission file saved successfully at:\n{submission_path}")
```

```
Submission file saved successfully at:
D:\ML Projects\Titanic2\submission.csv
```