In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
titanic_df = pd.read_csv("train.csv")
```

In [3]:
```
titanic_df.head(5)
```

Out[3]:

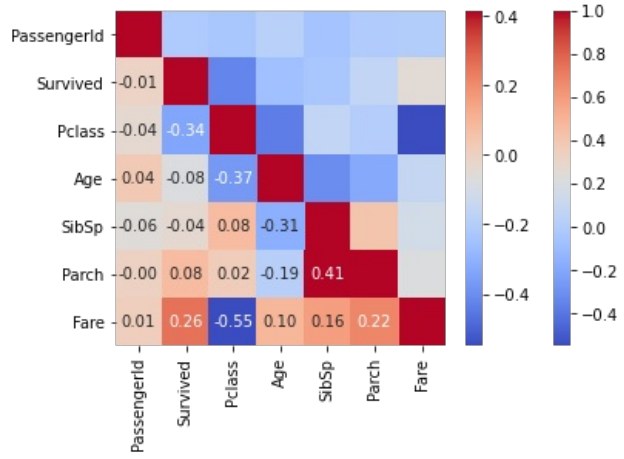
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 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

In [4]:
```
titanic_df.describe()
```

Out[4]:

	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

In [5]:
```
# Correlation matrix
corr = titanic_df.corr()

# Heatmap
sns.heatmap(corr,
            xticklabels=corr.columns,
            yticklabels=corr.columns,
            cmap='coolwarm')
mask = np.triu(np.ones_like(corr, dtype=bool))
sns.heatmap(corr, mask=mask, annot=True, fmt='.2f', cmap='coolwarm')

plt.show()
```

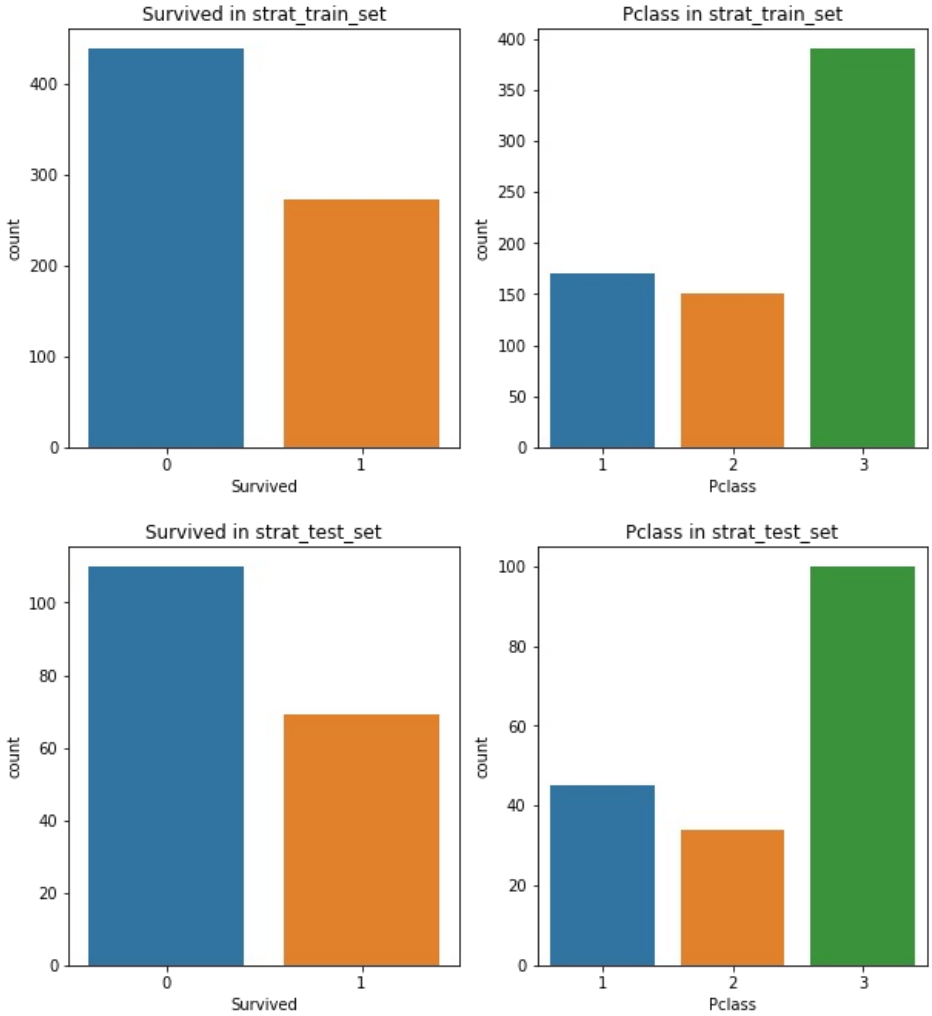


In [6]:
```
from sklearn.model_selection import StratifiedShuffleSplit
```

```
# Create a StratifiedShuffleSplit object
split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)

# Loop through the splits (we only have one)
for train_index, test_index in split.split(titanic_df, titanic_df['Survived'],titanic_df['Pclass']):
    # Get the training and testing data
    strat_train_set = titanic_df.iloc[train_index]
    strat_test_set = titanic_df.iloc[test_index]
```

In [7]:

```
# Create a figure and axes object using plt.subplots
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))

# Use seaborn to create a bar plot of the 'Survived' column on the first axes
sns.countplot(x='Survived', data=strat_train_set, ax=ax1)
ax1.set_title('Survived in strat_train_set')

# Use seaborn to create a bar plot of the 'Pclass' column on the second axes
sns.countplot(x='Pclass', data=strat_train_set, ax=ax2)
ax2.set_title('Pclass in strat_train_set')

# Create a figure and axes object using plt.subplots
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))

# Use seaborn to create a bar plot of the 'Survived' column on the first axes
sns.countplot(x='Survived', data=strat_test_set, ax=ax1)
ax1.set_title('Survived in strat_test_set')

# Use seaborn to create a bar plot of the 'Pclass' column on the second axes
sns.countplot(x='Pclass', data=strat_test_set, ax=ax2)
ax2.set_title('Pclass in strat_test_set')

plt.show()
```



In [8]:

```
strat_train_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 692 to 507
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
```

```
 0   PassengerId  712 non-null    int64
 1   Survived     712 non-null    int64
 2   Pclass       712 non-null    int64
 3   Name         712 non-null    object
 4   Sex          712 non-null    object
 5   Age          575 non-null    float64
 6   SibSp        712 non-null    int64
 7   Parch        712 non-null    int64
 8   Ticket       712 non-null    object
 9   Fare         712 non-null    float64
 10  Cabin        160 non-null    object
 11  Embarked     710 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 88.5+ KB
```

In [9]:
```
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.impute import SimpleImputer

class AgeImputer(BaseEstimator, TransformerMixin):

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        imputer = SimpleImputer(strategy="mean")
        X['Age'] = imputer.fit_transform(X[['Age']])
        return X
```

In [10]:
```
from sklearn.preprocessing import OneHotEncoder
class FeatureEncoder (BaseEstimator, TransformerMixin):

    def fit (self, X, y=None) :
        return self

    def transform (self, X):
        encoder = OneHotEncoder ()
        matrix = encoder.fit_transform(X [['Embarked']]).toarray ()
        column_names = ["C", "S", "Q", "N"]
        for i in range (len (matrix.T)):
            X[column_names[i]] = matrix.T[i]
        matrix = encoder.fit_transform(X[['Sex']]).toarray ()
        column_names = ["Female", "Male"]
        for i in range (len (matrix.T)):
            X[column_names[i]] = matrix.T[i]
        return X
```

In [11]:
```
class FeatureDropper(BaseEstimator, TransformerMixin):

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        return X.drop(["Embarked","Name","Ticket","Cabin","Sex","N"], axis=1, errors="ignore")
```

In [12]:
```
class FeatureDropper(BaseEstimator, TransformerMixin):
    def __init__(self, columns_to_drop):
        self.columns_to_drop = columns_to_drop

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        return X.drop(self.columns_to_drop, axis=1, errors="ignore")
```

In [13]:
```
fd = FeatureDropper(columns_to_drop = ["Embarked","Name","Ticket","Cabin","Sex","N"])
```

In [14]:
```
from sklearn.pipeline import Pipeline

pipeline = Pipeline([("ageimputer", AgeImputer()),
                     ("featureencoder", FeatureEncoder()),
                     ("featuredropper", fd)])
```

In [15]:
```
strat_train_set = pipeline.fit_transform(strat_train_set)
```

```
<ipython-input-9-bf47ab59dc18>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X['Age'] = imputer.fit_transform(X[['Age']])
<ipython-input-10-9b26102771b5>:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X[column_names[i]] = matrix.T[i]
<ipython-input-10-9b26102771b5>:16: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X[column_names[i]] = matrix.T[i]
```

In [16]:

```
strat_train_set
```

Out[16]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	C	S	Q	Female	Male
692	693	1	3	29.807687	0	0	56.4958	0.0	0.0	1.0	0.0	1.0
481	482	0	2	29.807687	0	0	0.0000	0.0	0.0	1.0	0.0	1.0
527	528	0	1	29.807687	0	0	221.7792	0.0	0.0	1.0	0.0	1.0
855	856	1	3	18.000000	0	1	9.3500	0.0	0.0	1.0	1.0	0.0
801	802	1	2	31.000000	1	1	26.2500	0.0	0.0	1.0	1.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...
359	360	1	3	29.807687	0	0	7.8792	0.0	1.0	0.0	1.0	0.0
258	259	1	1	35.000000	0	0	512.3292	1.0	0.0	0.0	1.0	0.0
736	737	0	3	48.000000	1	3	34.3750	0.0	0.0	1.0	1.0	0.0
462	463	0	1	47.000000	0	0	38.5000	0.0	0.0	1.0	0.0	1.0
507	508	1	1	29.807687	0	0	26.5500	0.0	0.0	1.0	0.0	1.0

712 rows × 12 columns

In [17]:

```
strat_train_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 692 to 507
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  712 non-null    int64  
 1   Survived     712 non-null    int64  
 2   Pclass       712 non-null    int64  
 3   Age          712 non-null    float64
 4   SibSp        712 non-null    int64  
 5   Parch        712 non-null    int64  
 6   Fare         712 non-null    float64
 7   C            712 non-null    float64
 8   S            712 non-null    float64
 9   Q            712 non-null    float64
 10  Female       712 non-null    float64
 11  Male         712 non-null    float64
dtypes: float64(7), int64(5)
memory usage: 88.5 KB
```

In [18]:

```
from sklearn.preprocessing import StandardScaler

X = strat_train_set.drop(['Survived'], axis =1)
y = strat_train_set['Survived']

scaler = StandardScaler().fit(X)
X_data = scaler.transform(X)
y_data = y.to_numpy()
```

In [19]:

In [19]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

clf = RandomForestClassifier()

param_gird =[
    {"n_estimators": [10,100,200,500], "max_depth": [None, 5, 10], "min_samples_split": [2,3,4]}
]

grid_search = GridSearchCV(clf, param_gird, cv=3, scoring = "accuracy", return_train_score=True)
grid_search.fit(X_data, y_data)
```

Out[19]:
```
GridSearchCV(cv=3, estimator=RandomForestClassifier(),
             param_grid=[{'max_depth': [None, 5, 10],
                          'min_samples_split': [2, 3, 4],
                          'n_estimators': [10, 100, 200, 500]}],
             return_train_score=True, scoring='accuracy')
```

In [20]:
```
final_clf = grid_search.best_estimator_
```

In [21]:
```
final_clf
```

Out[21]:
```
RandomForestClassifier(max_depth=10, min_samples_split=4, n_estimators=10)
```

In [22]:
```
strat_test_set = pipeline.fit_transform(strat_test_set)
```

```
<ipython-input-9-bf47ab59dc18>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X['Age'] = imputer.fit_transform(X[['Age']])
<ipython-input-10-9b26102771b5>:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X[column_names[i]] = matrix.T[i]
<ipython-input-10-9b26102771b5>:16: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X[column_names[i]] = matrix.T[i]
```

In [23]:
```
X_test = strat_test_set.drop(['Survived'], axis =1)
y_test = strat_test_set['Survived']

scaler = StandardScaler()
X_data_test = scaler.fit_transform(X_test)
y_data_test = y_test.to_numpy()
```

In [24]:
```
final_clf.score(X_data_test, y_data_test)
```

Out[24]:
```
0.7877094972067039
```

In [25]:
```
final_data = pipeline.fit_transform(titanic_df)
```

In [26]:
```
final_data
```

Out[26]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	C	S	Q	Female	Male
0	1	0	3	22.000000	1	0	7.2500	0.0	0.0	1.0	0.0	1.0
1	2	1	1	38.000000	1	0	71.2833	1.0	0.0	0.0	1.0	0.0

2	3	1	3	26.000000	0	0	7.9250	0.0	0.0	1.0	1.0	0.0
3	4	1	1	35.000000	1	0	53.1000	0.0	0.0	1.0	1.0	0.0
4	5	0	3	35.000000	0	0	8.0500	0.0	0.0	1.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	27.000000	0	0	13.0000	0.0	0.0	1.0	0.0	1.0
887	888	1	1	19.000000	0	0	30.0000	0.0	0.0	1.0	1.0	0.0
888	889	0	3	29.699118	1	2	23.4500	0.0	0.0	1.0	1.0	0.0
889	890	1	1	26.000000	0	0	30.0000	1.0	0.0	0.0	0.0	1.0
890	891	0	3	32.000000	0	0	7.7500	0.0	1.0	0.0	0.0	1.0

891 rows × 12 columns

In [29]:
```
X_final = final_data.drop(['Survived'], axis =1)
y_final = final_data['Survived']

scaler = StandardScaler()
X_data_final = scaler.fit_transform(X_final)
y_data_final = y_final.to_numpy()
```

In [30]:
```
production_clf = RandomForestClassifier()

param_gird =[
    {"n_estimators": [10,100,200,500], "max_depth": [None, 5, 10], "min_samples_split": [2,3,4]}
]

grid_search = GridSearchCV(production_clf, param_gird, cv=3, scoring = "accuracy", return_train_score=True)
grid_search.fit(X_data_final, y_data_final)
```

Out[30]:
```
GridSearchCV(cv=3, estimator=RandomForestClassifier(),
             param_grid=[{'max_depth': [None, 5, 10],
                          'min_samples_split': [2, 3, 4],
                          'n_estimators': [10, 100, 200, 500]}],
             return_train_score=True, scoring='accuracy')
```

In [31]:
```
production_final_clf = grid_search.best_estimator_
```

In [33]:
```
titanic_test_df = pd.read_csv("test.csv")
```

In [34]:
```
final_test_data = pipeline.fit_transform(titanic_test_df)
```

In [38]:
```
final_test_data
```

Out[38]:

	PassengerId	Pclass	Age	SibSp	Parch	Fare	C	S	Q	Female	Male
0	892	3	34.50000	0	0	7.8292	0.0	1.0	0.0	0.0	1.0
1	893	3	47.00000	1	0	7.0000	0.0	0.0	1.0	1.0	0.0
2	894	2	62.00000	0	0	9.6875	0.0	1.0	0.0	0.0	1.0
3	895	3	27.00000	0	0	8.6625	0.0	0.0	1.0	0.0	1.0
4	896	3	22.00000	1	1	12.2875	0.0	0.0	1.0	1.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...
413	1305	3	30.27259	0	0	8.0500	0.0	0.0	1.0	0.0	1.0
414	1306	1	39.00000	0	0	108.9000	1.0	0.0	0.0	1.0	0.0
415	1307	3	38.50000	0	0	7.2500	0.0	0.0	1.0	0.0	1.0
416	1308	3	30.27259	0	0	8.0500	0.0	0.0	1.0	0.0	1.0
417	1309	3	30.27259	1	1	22.3583	1.0	0.0	0.0	0.0	1.0

418 rows × 11 columns

In [39]:
```
final_test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Pclass       418 non-null    int64  
 2   Age          418 non-null    float64
 3   SibSp        418 non-null    int64  
 4   Parch        418 non-null    int64  
 5   Fare         417 non-null    float64
 6   C            418 non-null    float64
 7   S            418 non-null    float64
 8   Q            418 non-null    float64
 9   Female       418 non-null    float64
 10  Male         418 non-null    float64
dtypes: float64(7), int64(4)
memory usage: 36.0 KB
```

In [42]:
```
X_final_test = final_test_data
X_final_test = X_final_test.fillna(method="ffill")

scaler = StandardScaler()
X_data_final_test = scaler.fit_transform(X_final_test)
```

In [44]:
```
predictions = production_final_clf.predict(X_data_final_test)
```

In [45]:
```
predictions
```

Out[45]:
```
array([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, 1, 0, 0, 1, 1, 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, 1, 0, 0, 0, 0, 0, 1, 1,
       1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 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, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 1, 1, 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, 1, 0, 1, 1,
       0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 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, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 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, 1, 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],
      dtype=int64)
```

In [46]:
```
final_df = pd.DataFrame(titanic_test_df['PassengerId'])
final_df['Survived'] = predictions
final_df.to_csv("predictions.csv", index = False)
```

In [47]:
```
final_df
```

Out[47]:

	PassengerId	Survived
0	892	0
1	893	1
2	894	0
3	895	0
4	896	1
...	...	...
413	1305	0
414	1306	1
415	1307	0
416	1308	0
417	1309	0

418 rows × 2 columns

In [ ]: