

codsoft-ds-1

August 13, 2024

TASK-1 TITANIC SURVIVAL PREDICTION

```
[1]: import pandas as pd
      import numpy as np

      import matplotlib.pyplot as plt
      import plotly.express as px

      from sklearn.preprocessing import LabelEncoder
      from sklearn.impute import SimpleImputer
      from sklearn.model_selection import train_test_split, cross_val_score
      from sklearn.preprocessing import StandardScaler

      from sklearn.linear_model import LogisticRegression
      from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

      from sklearn.metrics import classification_report
```

```
[2]: titanic = pd.read_csv('/content/Titanic-Dataset.csv')
```

```
[3]: titanic.head()
```

```
[3]:   PassengerId  Survived  Pclass \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3

                                                Name     Sex   Age  SibSp \
0          Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                Heikkinen, Miss. Laina  female  26.0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4            Allen, Mr. William Henry    male  35.0      0

   Parch      Ticket     Fare Cabin Embarked
0     0    A/5 21171  7.2500   NaN       S
```

```
1      0          PC 17599  71.2833   C85      C
2      0  STON/O2. 3101282   7.9250    NaN      S
3      0          113803  53.1000   C123      S
4      0          373450  8.0500    NaN      S
```

```
[ ]: titanic.shape
```

```
[ ]: (418, 12)
```

```
[ ]: titanic.columns
```

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

```
[ ]: titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype  
---  --  
 0   PassengerId  418 non-null    int64  
 1   Survived     418 non-null    int64  
 2   Pclass       418 non-null    int64  
 3   Name         418 non-null    object 
 4   Sex          418 non-null    object 
 5   Age          332 non-null    float64
 6   SibSp        418 non-null    int64  
 7   Parch        418 non-null    int64  
 8   Ticket       418 non-null    object 
 9   Fare          417 non-null    float64
 10  Cabin         91 non-null    object 
 11  Embarked     418 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 39.3+ KB
```

```
[ ]: titanic.isna().sum()
```

```
[ ]: PassengerId      0
Survived          0
Pclass            0
Name              0
Sex               0
Age              86
SibSp            0
Parch            0
```

```

Ticket          0
Fare           1
Cabin         327
Embarked       0
dtype: int64

```

```
[ ]: # Dropping non essential coolumns
```

```
titanic.drop(['PassengerId', 'Name', 'Ticket'], axis = 1, inplace = True)
```

```
[ ]: titanic.head()
```

```
[ ]:   Survived Pclass      Sex   Age  SibSp  Parch     Fare Cabin Embarked
0        0      3    male  34.5      0      0    7.8292   NaN      Q
1        1      3  female  47.0      1      0    7.0000   NaN      S
2        0      2    male  62.0      0      0    9.6875   NaN      Q
3        0      3    male  27.0      0      0    8.6625   NaN      S
4        1      3  female  22.0      1      1   12.2875   NaN      S
```

```
[ ]: survived_counts = titanic['Survived'].value_counts()
fig_surv_perc = px.pie(titanic, names= survived_counts.index, values = ↴survived_counts.values, title=f'Distribution of Survived', hole=0.2, ↴color_discrete_sequence=px.colors.sequential.Viridis)
fig_surv_perc.update_traces(textinfo='percent+label')
fig_surv_perc.update_layout(legend_title_text='Categories:', ↴legend=dict(orientation="h", yanchor="bottom", y=1.02))
fig_surv_perc.show()
```

```
[ ]: fig_sex_count = px.histogram(titanic, x = 'Sex', color = 'Sex', ↴color_discrete_sequence=px.colors.sequential.Viridis)
fig_sex_count.update_layout(title_text='Count of different Sex', ↴xaxis_title='Sex', yaxis_title='Count', plot_bgcolor = 'white')
fig_sex_count.show()

fig_sex_perc = px.pie(titanic, names= 'Sex', title=f'Distribution of Sex', ↴hole=0.2, color_discrete_sequence=px.colors.sequential.Viridis)
fig_sex_perc.update_traces(textinfo='percent+label')
fig_sex_perc.update_layout(legend_title_text='Categories:', ↴legend=dict(orientation="h", yanchor="bottom", y=1.02))
fig_sex_perc.show()
```

```
[ ]: fig_pclass_surv = px.histogram(titanic, x = 'Sex', barmode = 'group', color = ↴'Survived', color_discrete_sequence=px.colors.sequential.Viridis)
fig_pclass_surv.update_layout(title = 'Survival according to gender', ↴plot_bgcolor = 'white')
fig_pclass_surv.show()
```

```
[ ]: import pandas as pd
      from sklearn.preprocessing import LabelEncoder

      le = LabelEncoder()
      cols = ['Sex', 'Cabin', 'Embarked'] # Removed 'Title' as it seems to be missing

      for col in cols:
          titanic[col] = le.fit_transform(titanic[col])

[ ]: # Checking the class count for target variable

      titanic.Survived.value_counts()

[ ]: Survived
      0    266
      1    152
      Name: count, dtype: int64

[ ]: # Handle missing values before scaling (replace with mean for example)
      imputer = SimpleImputer(strategy='mean')
      X = titanic.drop('Survived', axis=1)
      X_imputed = pd.DataFrame(imputer.fit_transform(X), columns=X.columns)

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

[ ]: # Splitting the dataset into training and testing parts

      X_train, X_test, y_train, y_test = train_test_split(X_imputed,y, test_size = 0.
      ↪3, random_state = 42)

[ ]: # Doing feature scaling by StandardScaler

      sc = StandardScaler()
      X_train_scaled = sc.fit_transform(X_train)
      X_test_scaled = sc.transform(X_test)

[ ]: lr = LogisticRegression()
      lr.fit(X_train_scaled, y_train)
      y_pred_lr = lr.predict(X_test_scaled)

[ ]: lr_report = classification_report(y_test, y_pred_lr) # Use y_pred_lr instead of ↪lr_pred
      lr_scores = cross_val_score(lr, X_train_scaled, y_train, cv=5, ↪scoring='accuracy')
      print(lr_report)
      print("Cross validation scores:", lr_scores)
```

```
lr_scores = cross_val_score(lr, X_train_scaled, y_train, cv=5,  
                           scoring='accuracy')
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	85
1	1.00	1.00	1.00	41
accuracy			1.00	126
macro avg	1.00	1.00	1.00	126
weighted avg	1.00	1.00	1.00	126

Cross validation scores: [1. 1. 1. 1. 1.]

```
[ ]: import seaborn as sns  
import matplotlib.pyplot as plt  
import pandas as pd  
from sklearn.metrics import classification_report  
  
# Generate classification report  
lr_report = classification_report(y_test, y_pred_lr, output_dict=True)  
  
# Convert classification report to DataFrame  
report_df = pd.DataFrame(lr_report).transpose()  
  
# Plot heatmap  
plt.figure(figsize=(10, 6))  
sns.heatmap(report_df.iloc[:-1, :-1].astype(float), annot=True, cmap='Blues',  
            fmt='.2f', linewidths=.5)  
plt.title('Classification Report Heatmap')  
plt.show()
```

