Muhammad Umer Adeeb

Question2: Titanic

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv('/content/titanic.csv')
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 891,\n  \"fields\": [\n    {\n      \"column\": \"PassengerId\",\n      \"properties\": {\n\"dtype\": \"number\",\n        \"std\": 257,\n        \"min\": 1,\n\"max\": 891,\n        \"num_unique_values\": 891,\n\"samples\": [\n          710,\n          440,\n          841\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,\n0\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,\n1\n        ],\n        \"semantic_type\": \"\",\n\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Name\",\n      \"properties\": {\n        \"dtype\": \"string\",\n\"num_unique_values\": 891,\n        \"samples\": [\n\"Moubarek, Master. Halim Gonios (\\\"William George\\\")\",\n\"Kvillner, Mr. Johan Henrik Johannesson\"\n        ],\n\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Sex\",\n      \"properties\": {\n\"dtype\": \"category\",\n        \"num_unique_values\": 2,\n\"samples\": [\n          \"female\",\n          \"male\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n}\n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 14.526497332334044,\n        \"min\": 0.42,\n        \"max\": 80.0,\n\"num_unique_values\": 88,\n        \"samples\": [\n          0.75,\n22.0\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,\n0\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\": 681,\n   \"samples\": [\n
\"11774\",\n   \"248740\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Fare\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 49.693428597180905,\n
\"min\": 0.0,\n   \"max\": 512.3292,\n
\"num_unique_values\": 248,\n   \"samples\": [\n
11.2417,\n   51.8625\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Cabin\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 147,\n
\"samples\": [\n   \"D45\",\n   \"B49\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Embarked\",\n   \"properties\":
{\n   \"dtype\": \"category\",\n   \"num_unique_values\":
3,\n   \"samples\": [\n   \"S\",\n   \"C\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe","variable_name":"df"}

```
df.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
```

```
df.describe()
```

{"summary":"{\n \"name\": \"df\",\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,\n668.5\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,\n0.4865924542648585\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,\n3.0\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\": [\n29.69911764705882,\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,\n0.5230078563411896,\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,\n6.0,\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,\n14.4542,\n   891.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
df.shape
```

```
(891, 12)
```

```
df.isnull().sum()
```

```
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
```

```
df.drop(columns=['PassengerId', 'Name', 'Ticket', 'Cabin'],
inplace=True)
```

```
df.head()
```

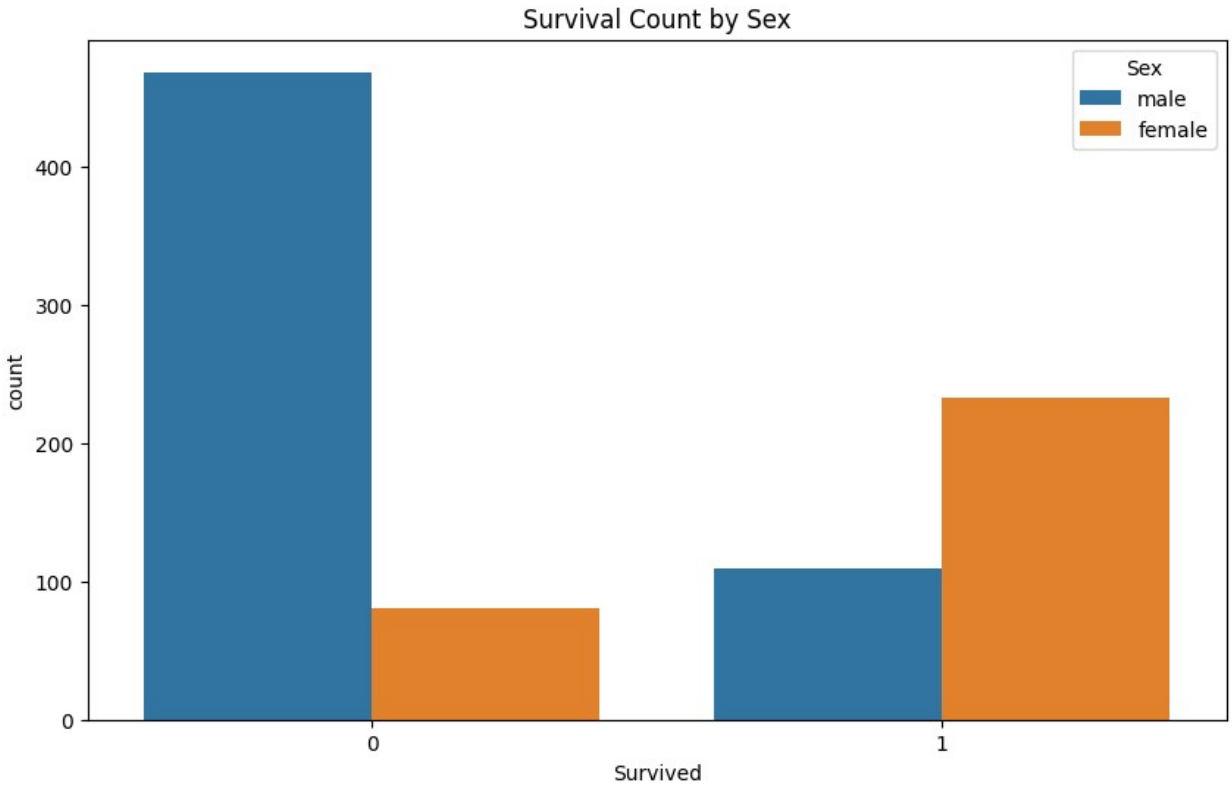

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 891,\n  \"fields\": [\
n    {\n      \"column\": \"Survived\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          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\": \"Sex\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"female\",\n          \"male\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 14.526497332334044,\n
\"min\": 0.42,\n        \"max\": 80.0,\n        \"num_unique_values\":
88,\n        \"samples\": [\n          0.75,\n          22.0\
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\":
```

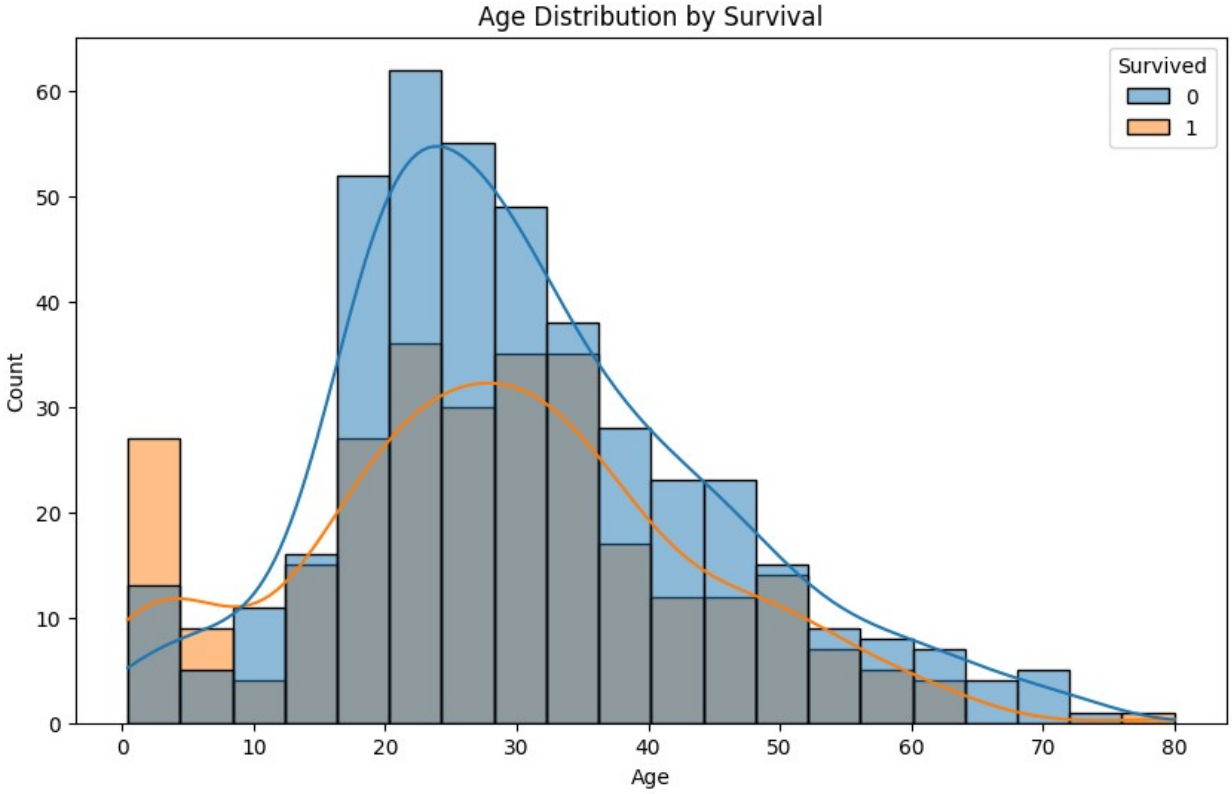
\"Fare\",\n   \"properties\": {\n   \"dtype\": \"number\",\n \"std\": 49.693428597180905,\n   \"min\": 0.0,\n   \"max\": 512.3292,\n   \"num_unique_values\": 248,\n   \"samples\": [\n   11.2417,\n   51.8625\n   ],\n \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Embarked\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 3,\n   \"samples\": [\n   \"S\",\n   \"C\"\n],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n }\n   }\n]\n}","type":"dataframe","variable_name":"df"}

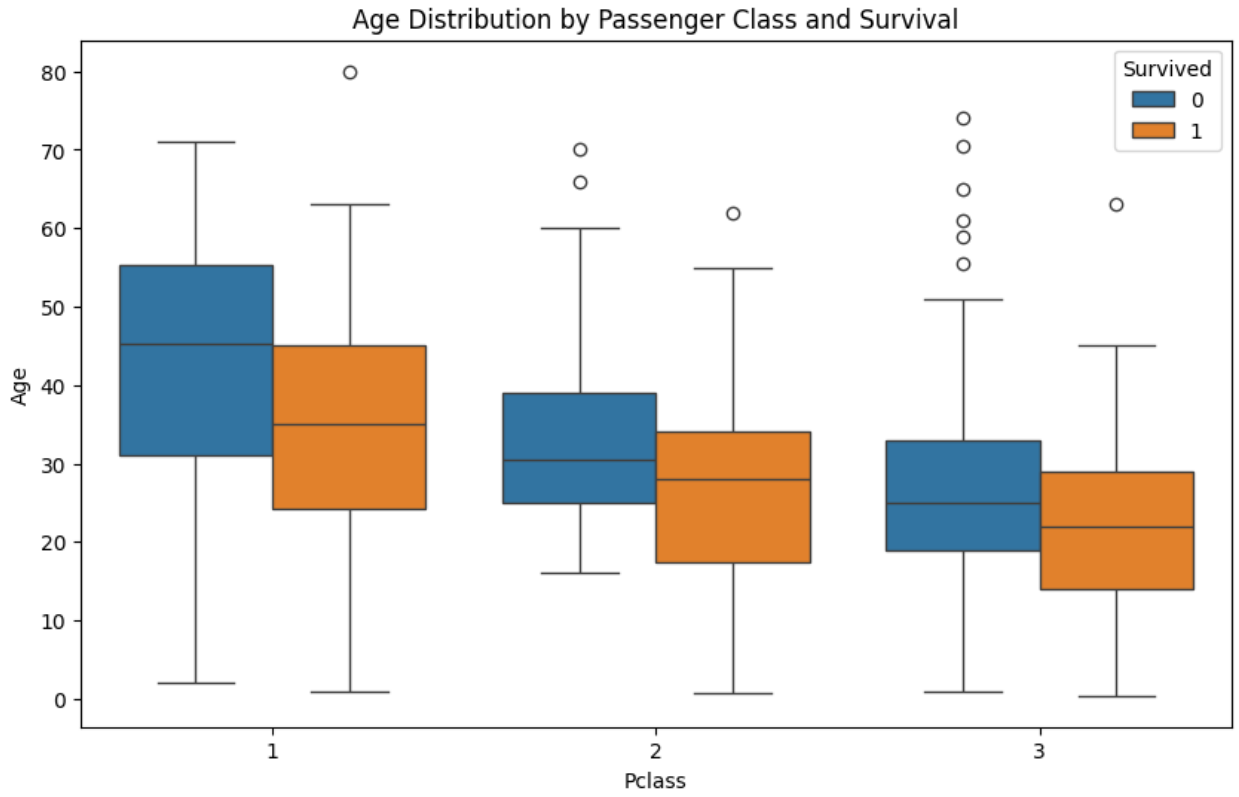
```
plt.figure(figsize=(10, 6))
sns.countplot(x='Survived', hue='Sex', data=df)
plt.title('Survival Count by Sex')
plt.show()

plt.figure(figsize=(10, 6))
sns.histplot(x='Age', hue='Survived', data=df, kde=True)
plt.title('Age Distribution by Survival')
plt.show()


plt.figure(figsize=(10, 6))
sns.boxplot(x='Pclass', y='Age', hue='Survived', data=df)
plt.title('Age Distribution by Passenger Class and Survival')
plt.show()
```


Survival Count by Sex
Sex
male
female
count
400
300
200
100
0
0
1
Survived

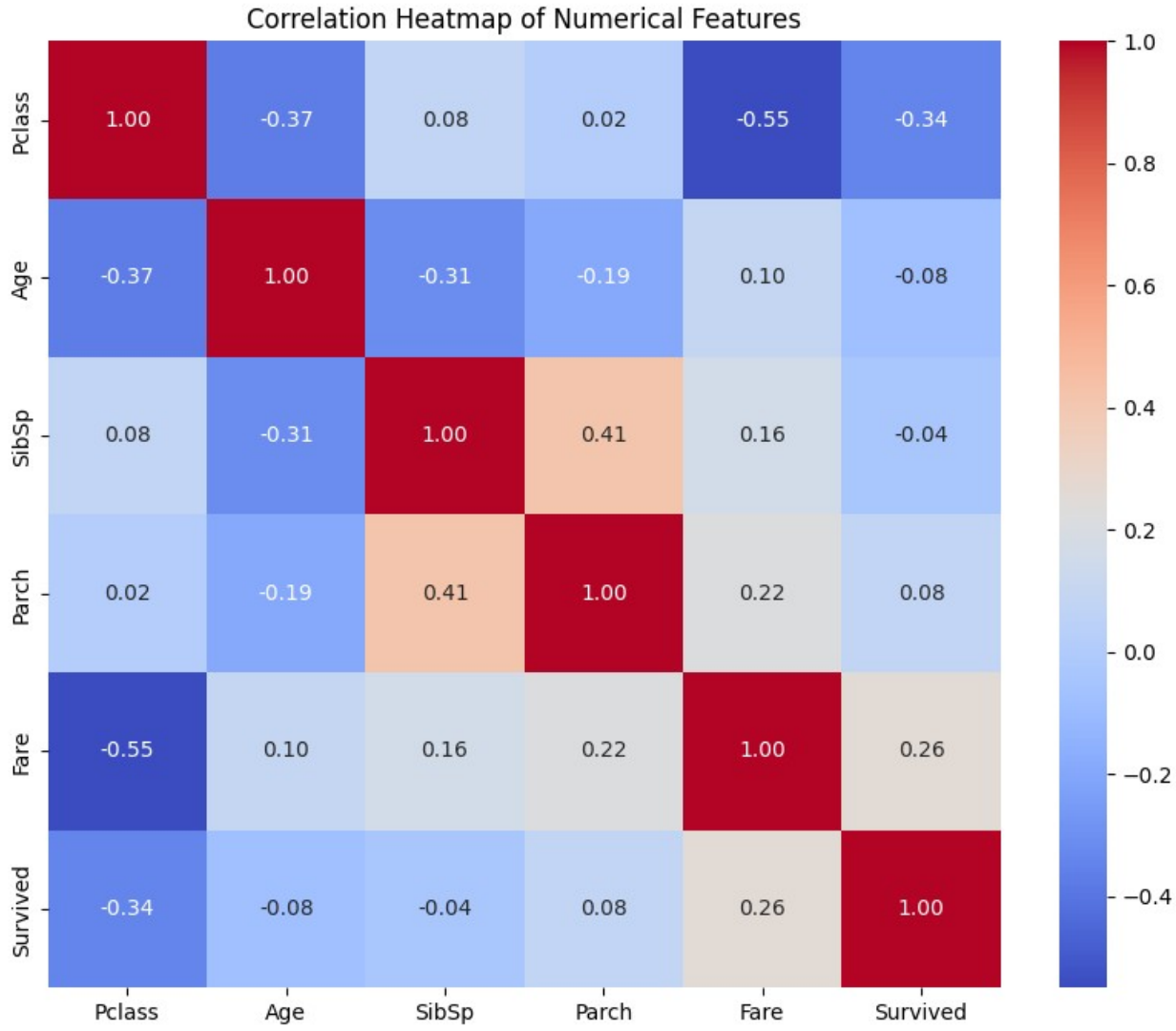

Age Distribution by Survival
Survived
0
1
Count
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
Age




```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Sex       891 non-null    object
 3   Age       714 non-null    float64
 4   SibSp     891 non-null    int64
 5   Parch     891 non-null    int64
 6   Fare      891 non-null    float64
 7   Embarked  889 non-null    object
dtypes: float64(2), int64(4), object(2)
memory usage: 55.8+ KB
```

```
numerical_cols = ['Pclass', 'Age', 'SibSp', 'Parch', 'Fare',
'Survived']
plt.figure(figsize=(10, 8))
sns.heatmap(df[numerical_cols].corr(), annot=True, cmap='coolwarm',
fmt=".2f")
plt.title('Correlation Heatmap of Numerical Features')
plt.show()
```




```
x = df.drop(columns=['Survived'])
y= df['Survived']
display(x.head())
display(y.head())
```


```
{"summary":"{\n  \"name\": \"display(y\",\n  \"rows\": 5,\n
\"fields\": [\n    {\n      \"column\": \"Pclass\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1,\n        \"min\": 1,\n        \"max\": 3,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1,\n
3\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Sex\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n
\"female\",\n          \"male\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n
```

\"dtype\": \"number\",\n   \"std\": 6.833739825307955,\n
\"min\": 22.0,\n   \"max\": 38.0,\n   \"num_unique_values\":
4,\n   \"samples\": [\n   38.0,\n   35.0\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"SibSp\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n
\"num_unique_values\": 2,\n   \"samples\": [\n   0,\n
1\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\": 0,\n
\"num_unique_values\": 1,\n   \"samples\": [\n   0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Fare\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\": 30.5100288352535,\
n   \"min\": 7.25,\n   \"max\": 71.2833,\n
\"num_unique_values\": 5,\n   \"samples\": [\n   71.2833\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Embarked\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n   \"samples\": [\n   \"C\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe"}

```
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

```
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=45)
```

```
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

```
print(x.isnull().sum())
print(x.columns)
```

```
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
Fare          0
Embarked      2
dtype: int64
Index(['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked'],
dtype='object')
```

```
trf1 = ColumnTransformer([
    ('impute_age', SimpleImputer(),[2]),
    ('impute_embarked', SimpleImputer(strategy='most_frequent'),[6])
], remainder='passthrough')
```

```
trf2 = ColumnTransformer([
    ('ohe_sex_embarked', OneHotEncoder(sparse_output=False,
handle_unknown='ignore'),[1,6])
],remainder='passthrough')
```

```
df['Sex'].nunique()
```

```
2
```

```
df['Embarked'].nunique()
```

```
3
```

```
trf3 = ColumnTransformer([
    ('scale', MinMaxScaler(),slice(0,10))
])
```

```
trf4 = SelectKBest(score_func=chi2, k=6)
```

```
trf5 = RandomForestClassifier()
```

```
pipe = Pipeline([
                ('trf1', trf1),
                ('trf2', trf2),
                ('trf3', trf3),
                ('trf4', trf4),
                ('trf5', trf5)
])
```

```
pipe
```

```
Pipeline(steps=[('trf1',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('impute_age',
SimpleImputer(),
                                                  [2]),
```

```
                                                  ('impute_embarked',

SimpleImputer(strategy='most_frequent'),
                                                   [6])])),
                ('trf2',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('ohe_sex_embarked',

OneHotEncoder(handle_unknown='ignore',

sparse_output=False),
                                                   [1, 6])])),
                ('trf3',
                 ColumnTransformer(transformers=[('scale',
MinMaxScaler(),
                                                   slice(0, 10,
None))])),
                ('trf4',
                 SelectKBest(k=6,
                             score_func=<function chi2 at
0x7af809da3380>)),
                ('trf5', RandomForestClassifier())])

pipe.named_steps

{'trf1': ColumnTransformer(remainder='passthrough',
                   transformers=[('impute_age', SimpleImputer(), [2]),
                                 ('impute_embarked',

SimpleImputer(strategy='most_frequent'),
                                  [6])]),
 'trf2': ColumnTransformer(remainder='passthrough',
                   transformers=[('ohe_sex_embarked',

OneHotEncoder(handle_unknown='ignore',
                                                  sparse_output=False),
                                  [1, 6])]),
 'trf3': ColumnTransformer(transformers=[('scale', MinMaxScaler(),
slice(0, 10, None))]),
 'trf4': SelectKBest(k=6, score_func=<function chi2 at
0x7af809da3380>),
 'trf5': RandomForestClassifier()}

pipe.fit(x_train, y_train)

Pipeline(steps=[('trf1',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('impute_age',
SimpleImputer(),
                                                  [2]),
                                                  ('impute_embarked',
```

```
SimpleImputer(strategy='most_frequent'),
                                                  [6])])),
                ('trf2',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('ohe_sex_embarked',

OneHotEncoder(handle_unknown='ignore',

sparse_output=False),
                                                  [1, 6])])),
                ('trf3',
                 ColumnTransformer(transformers=[('scale',
MinMaxScaler(),
                                                  slice(0, 10,
None))])),
                ('trf4',
                 SelectKBest(k=6,
                             score_func=<function chi2 at
0x7af809da3380>)),
                ('trf5', RandomForestClassifier())])
```

```
y_pred = pipe.predict(x_test)
y_pred
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0,
1,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
1,
       1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0,
0,
       0, 0, 0])
```

```
pd.DataFrame({'y_test':y_test, 'y_predict':y_pred})
```


```
{"summary":"{\n  \"name\": \"pd\",\n  \"rows\": 179,\n  \"fields\": [\
n    {\n      \"column\": \"y_test\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
```

```
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"y_predict\",\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}","type":"dataframe"}
```

```
print('Accuracy:', accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7318435754189944
```

```
from sklearn.model_selection import cross_val_score
cross_val_score(pipe, x_train, y_train, cv=5,
scoring='accuracy').mean()
```

```
0.587067861715749
```

```
params = {
    'trf5__max_depth':[1,2,3,4,5,None]
}
```

```
from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(pipe, params, cv=5, scoring='accuracy')
grid.fit(x_train, y_train)
```

```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('trf1',

ColumnTransformer(remainder='passthrough',

transformers=[('impute_age',

SimpleImputer(),

[2]),

('impute_embarked',

SimpleImputer(strategy='most_frequent'),

[6])])),
                                       ('trf2',

ColumnTransformer(remainder='passthrough',

transformers=[('ohe_sex_embarked',

OneHotEncoder(handle_unknown='ignore',

sparse_output=False),
```

```
[1,

6])])),
                                          ('trf3',

ColumnTransformer(transformers=[('scale',

MinMaxScaler(),

slice(0, 10, None))])),
                                          ('trf4',
                                           SelectKBest(k=6,

score_func=<function chi2 at 0x7af809da3380>)),
                                          ('trf5',
RandomForestClassifier())]),
             param_grid={'trf5__max_depth': [1, 2, 3, 4, 5, None]},
             scoring='accuracy')

grid.best_score_

0.6067369250467842
```

Question: 3 Covid Test

```
sample_size = 500
true_positive = 45
false_positive = 55
false_negative = 5
true_negative = 395

print('Accuracy:', ((45+395)/500*100))
print('Precision:', (45/(45+55)*100))
print('Recall:', (45/(45+5)*100))
print('F1 Score:', (2*45/(2*45+55+5))*100)

confusion_matrix = np.array([[true_positive, false_positive],
                             [false_negative, true_negative]])
df_cm = pd.DataFrame(confusion_matrix,
                     index=['Actual Positive', 'Actual Negative'],
                     columns=['Predicted Positive', 'Predicted
Negative'])
display("Confusion Matrix:")
print(df_cm)

Accuracy: 88.0
Precision: 45.0
Recall: 90.0
F1 Score: 60.0
```

```
{"type":"string"}
```

```
                  Predicted Positive  Predicted Negative
Actual Positive                   45                  55
Actual Negative                    5                 395
```

Question 5: Definitions

Underfitting

A statistical model or a machine learning algorithm is said to have underfitting when a model is too simple to capture data complexities. It represents the inability of the model to learn the training data effectively result in poor performance both on the training and testing data. In simple terms, an underfit model's are inaccurate, especially when applied to new, unseen examples. It mainly happens when we uses very simple model with overly simplified assumptions. To address underfitting problem of the model, we need to use more complex models, with enhanced feature representation.

Overfitting

A statistical model is said to be overfitted when the model does not make accurate predictions on testing data. Overfitting is a problem where the evaluation of machine learning algorithms on training data is different from unseen data.

Best fit

A statistical model is said to be the best fit when it generalizes well to unseen data. A well-fitted model maintains a balance between training accuracy and performance on testing data, avoiding both overfitting and underfitting.

Bias

Bias refers to the error due to overly simplistic assumptions in the learning algorithm. These assumptions make the model easier to comprehend and learn but might not capture the underlying complexities of the data. It is the error due to the model's inability to represent the true relationship between input and output accurately. When a model has poor performance both on the training and testing data means high bias because of the simple model, indicating underfitting.