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
```

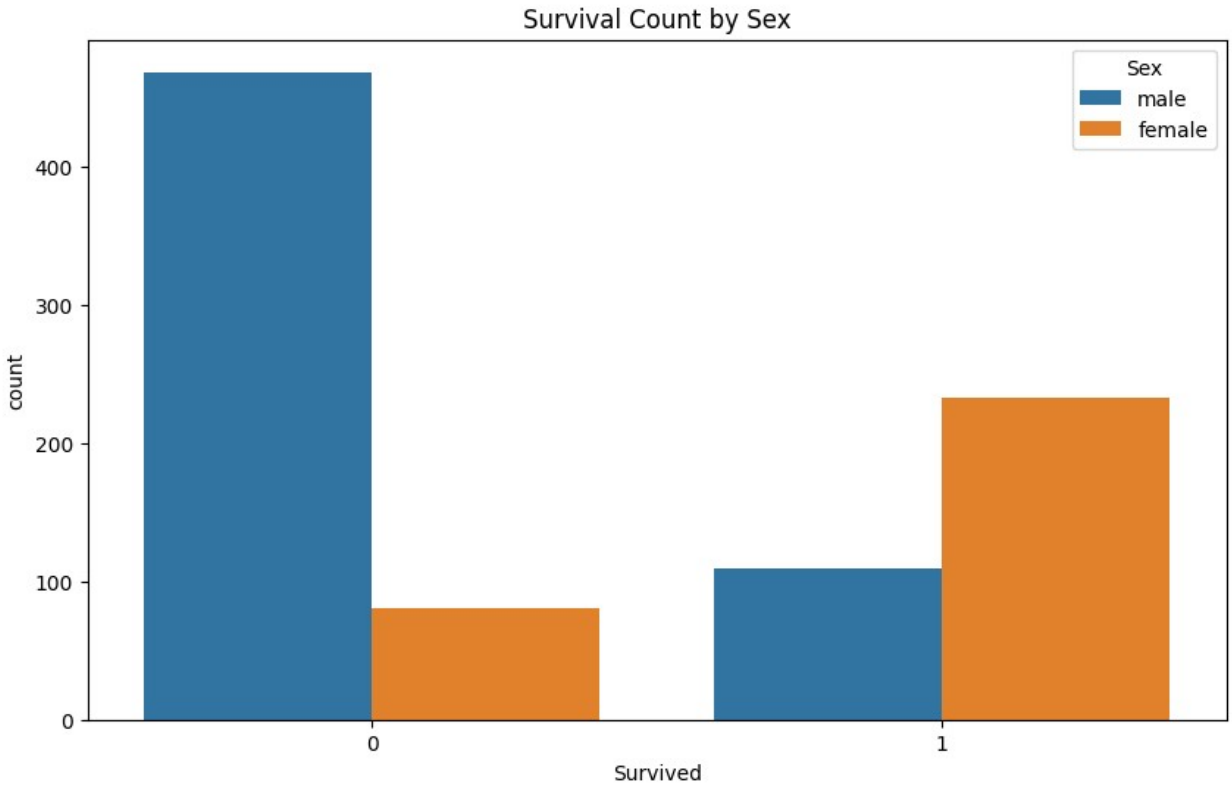
```
df.drop(columns=['PassengerId', 'Name', 'Ticket', 'Cabin'],
inplace=True)
```

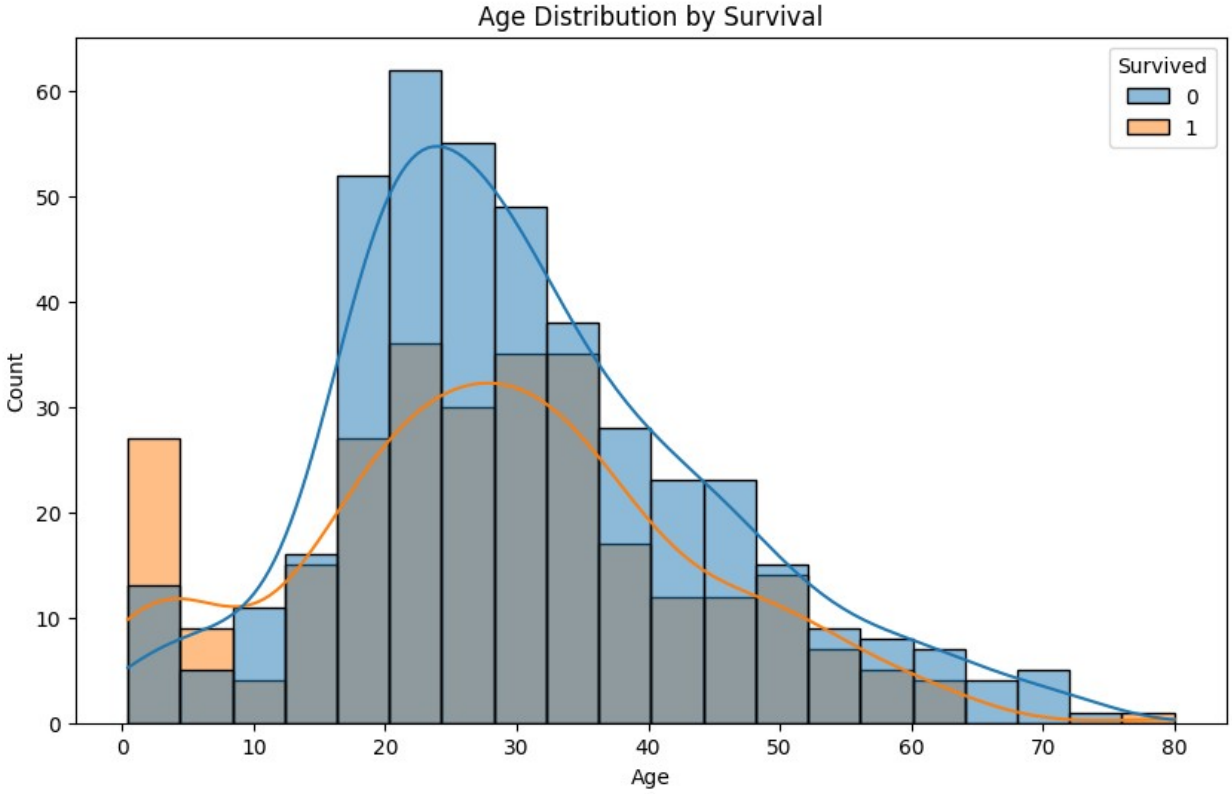
```
df.head()
```

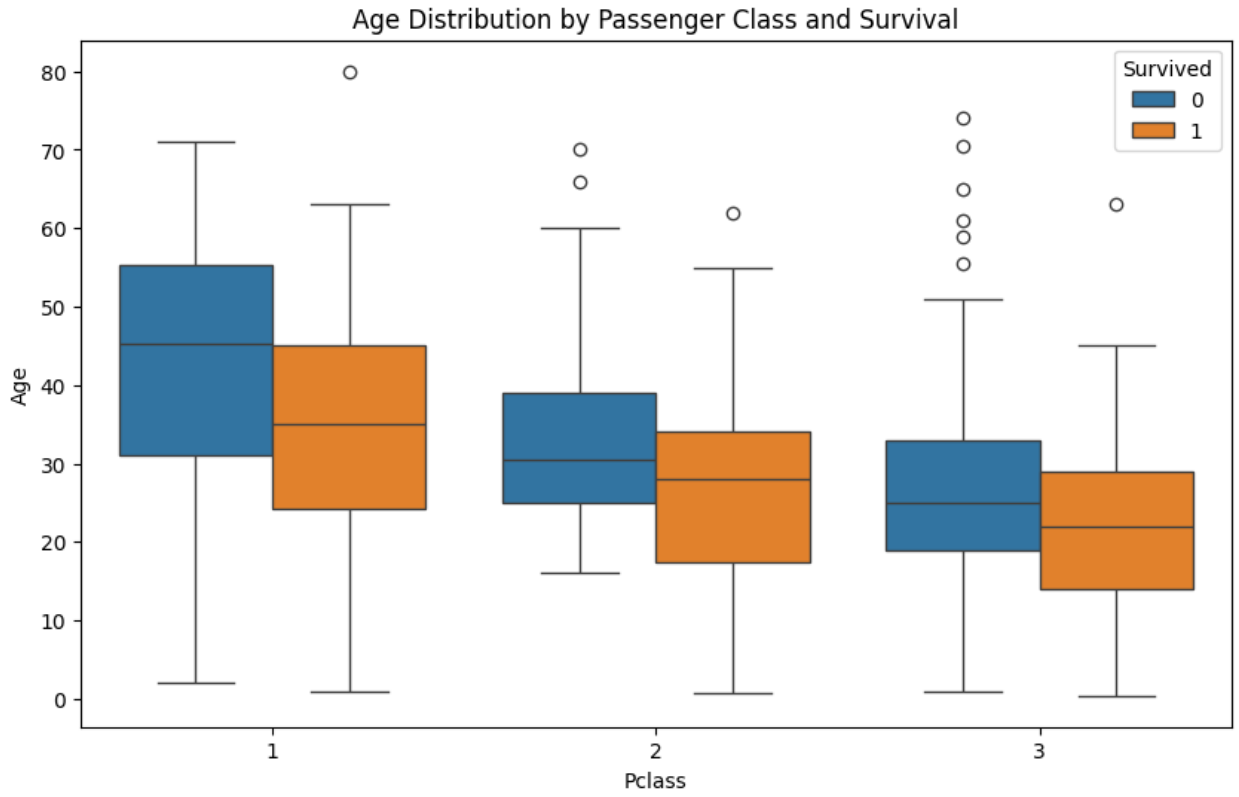

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 891,\n  \"fields\": [\
n    {\n      \"column\": \"Survived\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          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

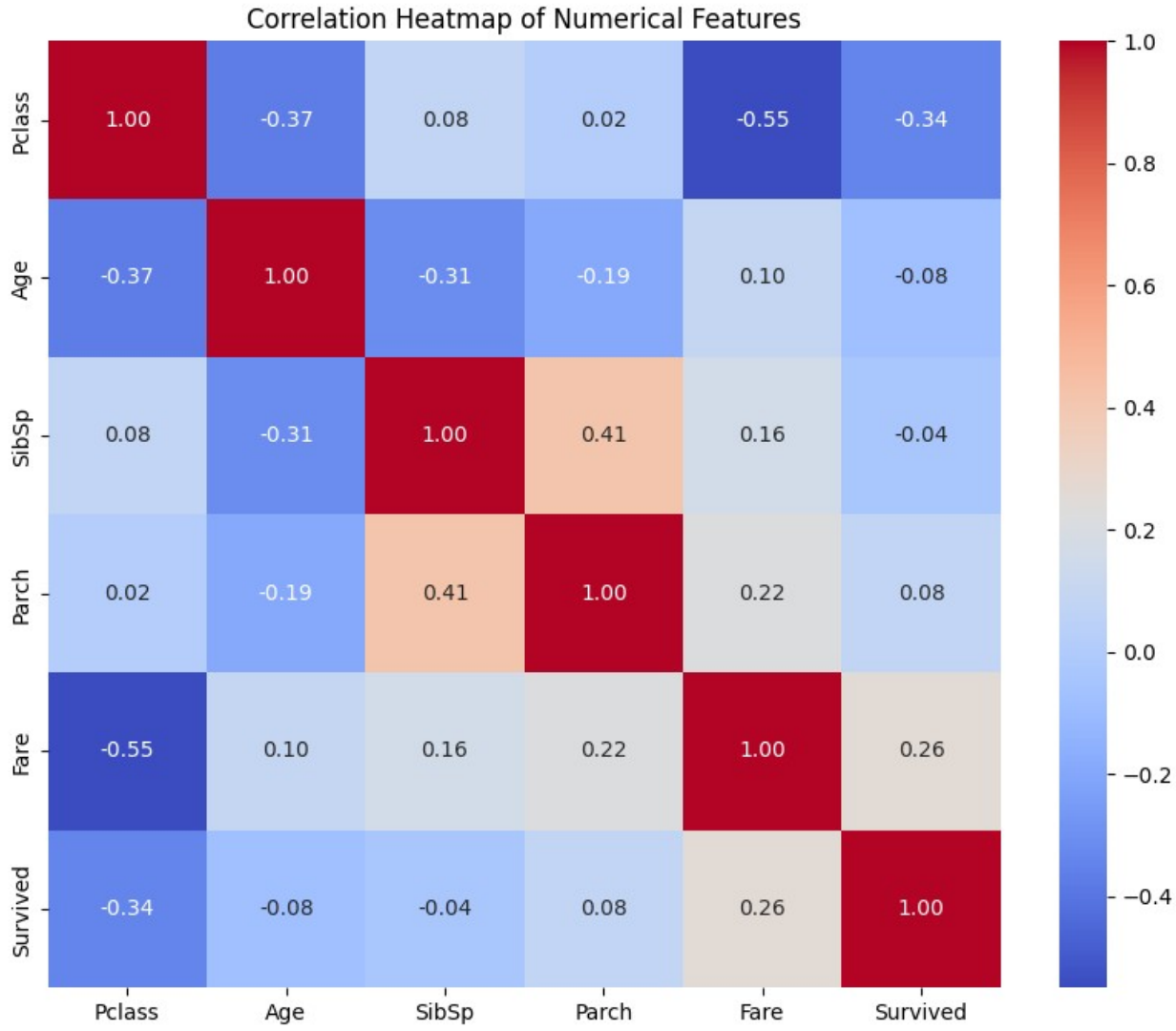

Age Distribution by Survival
Survived
0
1
Count
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
Age




```
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.

Muhammad Umer Adeeb

Question 4: Kaggle Dataset

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

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


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 303,\n  \"fields\": [\
n    {\n      \"column\": \"Unnamed: 0\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 87,\n        \"min\": 1,\n
\"max\": 303,\n        \"num_unique_values\": 303,\n
\"samples\": [\n          180,\n          229,\n          112\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 9,\n        \"min\":
29,\n        \"max\": 77,\n        \"num_unique_values\": 41,\n
\"samples\": [\n          61,\n          64,\n          44\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Sex\",\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\":
\"ChestPain\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          \"asymptomatic\",\n          \"nontypical\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"RestBP\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 17,\n
\"min\": 94,\n        \"max\": 200,\n        \"num_unique_values\":
50,\n        \"samples\": [\n          124,\n          192\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Chol\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 51,\n        \"min\": 126,\n        \"max\": 564,\n
\"num_unique_values\": 152,\n        \"samples\": [\n          321,\n
187\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Fbs\",\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\":
```

```
\"RestECG\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 0,\n        \"min\": 0,\n        \"max\": 2,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          2,\n
0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"MaxHR\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 22,\n        \"min\": 71,\n        \"max\": 202,\n
\"num_unique_values\": 91,\n        \"samples\": [\n          170,\n
114\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"ExAng\",\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\":
\"Oldpeak\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 1.161075022068634,\n        \"min\": 0.0,\n
\"max\": 6.2,\n        \"num_unique_values\": 40,\n
\"samples\": [\n          2.4,\n          0.2\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Slope\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 0,\n        \"min\":
1,\n        \"max\": 3,\n        \"num_unique_values\": 3,\n
\"samples\": [\n          3,\n          2\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Ca\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.9374383177242157,\n
\"min\": 0.0,\n        \"max\": 3.0,\n        \"num_unique_values\":
4,\n        \"samples\": [\n          3.0,\n          1.0\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Thal\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
3,\n        \"samples\": [\n          \"fixed\",\n
\"normal\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"AHD\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          \"Yes\",\
n          \"No\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  303 non-null    int64
 1   Age         303 non-null    int64
```

```
 2   Sex          303 non-null    int64
 3   ChestPain    303 non-null    object
 4   RestBP       303 non-null    int64
 5   Chol         303 non-null    int64
 6   Fbs          303 non-null    int64
 7   RestECG      303 non-null    int64
 8   MaxHR        303 non-null    int64
 9   ExAng        303 non-null    int64
 10  Oldpeak      303 non-null    float64
 11  Slope        303 non-null    int64
 12  Ca           299 non-null    float64
 13  Thal         301 non-null    object
 14  AHD          303 non-null    object
dtypes: float64(2), int64(10), object(3)
memory usage: 35.6+ KB
```

```
df.shape
```

```
(303, 15)
```

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

```
Unnamed: 0    0
Age           0
Sex           0
ChestPain     0
RestBP        0
Chol          0
Fbs           0
RestECG       0
MaxHR         0
ExAng         0
Oldpeak       0
Slope         0
Ca            4
Thal          2
AHD           0
dtype: int64
```

```
df.dtypes
```

```
Unnamed: 0      int64
Age             int64
Sex             int64
ChestPain      object
RestBP          int64
Chol            int64
Fbs             int64
RestECG         int64
MaxHR           int64
ExAng           int64
```

```
Oldpeak       float64
Slope           int64
Ca            float64
Thal           object
AHD            object
dtype: object
```

```
for col in df.columns:
    zero_count = (df[col] == 0).sum()
    print(f"Number of zeros in {col}: {zero_count}")
```

```
Number of zeros in Unnamed: 0: 0
Number of zeros in Age: 0
Number of zeros in Sex: 97
Number of zeros in ChestPain: 0
Number of zeros in RestBP: 0
Number of zeros in Chol: 0
Number of zeros in Fbs: 258
Number of zeros in RestECG: 151
Number of zeros in MaxHR: 0
Number of zeros in ExAng: 204
Number of zeros in Oldpeak: 99
Number of zeros in Slope: 0
Number of zeros in Ca: 176
Number of zeros in Thal: 0
Number of zeros in AHD: 0
```

```
df['Age'].mean()
```

```
54.43894389438944
```

```
df.columns
```

```
Index(['Unnamed: 0', 'Age', 'Sex', 'ChestPain', 'RestBP', 'Chol',
'Fbs',
       'RestECG', 'MaxHR', 'ExAng', 'Oldpeak', 'Slope', 'Ca', 'Thal',
'AHD'],
      dtype='object')
```

```
#Now extract only Age, Sex, ChestPain, RestBP, Chol.
df.drop(columns=['Unnamed: 0', 'Fbs','RestECG', 'MaxHR', 'ExAng',
'Oldpeak', 'Slope', 'Ca', 'Thal'], inplace=True)
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 303,\n  \"fields\": [\
n    {\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 9,\n        \"min\": 29,\n
\"max\": 77,\n        \"num_unique_values\": 41,\n        \"samples\":
[\n          61,\n          64,\n          44\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Sex\",\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\": \"ChestPain\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 4,\n   \"samples\": [\n   \"asymptomatic\",\n   \"nontypical\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"RestBP\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 17,\n   \"min\": 94,\n   \"max\": 200,\n   \"num_unique_values\": 50,\n   \"samples\": [\n   124,\n   192\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Chol\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 51,\n   \"min\": 126,\n   \"max\": 564,\n   \"num_unique_values\": 152,\n   \"samples\": [\n   321,\n   187\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"AHD\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"Yes\",\n   \"No\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"df"}

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

{"summary":"{\n \"name\": \"display(y\",\n \"rows\": 5,\n \"fields\": [\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 14,\n   \"min\": 37,\n   \"max\": 67,\n   \"num_unique_values\": 4,\n   \"samples\": [\n   67,\n   41,\n   63\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Sex\",\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\": \"ChestPain\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 4,\n   \"samples\": [\n   \"asymptomatic\",\n   \"nontypical\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"RestBP\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 15,\n   \"min\": 120,\n   \"max\": 160,\n   \"num_unique_values\": 4,\n   \"samples\": [\n   160,\n   130\n   ],\

n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Chol\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 30,\n   \"min\": 204,\n   \"max\": 286,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   286,\n   204\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
0     No
1    Yes
2    Yes
3     No
4     No
Name: AHD, dtype: object
```

```
y.replace({'Yes': 1, 'No': 0}, inplace=True)
display(y.head())
```

```
<ipython-input-12-a8acbb79871c>:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  y.replace({'Yes': 1, 'No': 0}, inplace=True)
```

```
0    0
1    1
2    1
3    0
4    0
Name: AHD, 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=0.25, random_state=90)
print(x_train.shape)
display(x_train.head())
print(x_test.shape)
display(x_test.head())
print(y_train.shape)
display(y_train.head())
print(y_test.shape)
display(y_test.head)
```

```
(227, 5)
```

{"summary":"{\n \"name\": \"display(y_test\",\n \"rows\": 5,\n \"fields\": [\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 8,\n   \"min\": 40,\n   \"max\": 62,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   59,\n

```
55,\n          57\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Sex\",\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\":
\"ChestPain\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          \"asymptomatic\",\n          \"typical\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"RestBP\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 19,\n
\"min\": 132,\n        \"max\": 178,\n        \"num_unique_values\":
4,\n        \"samples\": [\n          178,\n          140\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Chol\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 57,\n
\"min\": 199,\n        \"max\": 342,\n        \"num_unique_values\":
5,\n        \"samples\": [\n          270,\n          342\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```


```
(76, 5)
```


```
{"summary":"{\n  \"name\": \"display(y_test\",\n  \"rows\": 5,\n
\"fields\": [\n    {\n      \"column\": \"Age\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3,\n        \"min\": 53,\n        \"max\": 61,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          53,\n
59,\n          56\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Sex\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 1,\n        \"max\": 1,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          1\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"ChestPain\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n
\"asymptomatic\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"RestBP\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 11,\n        \"min\": 110,\n        \"max\": 138,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          123\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Chol\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 46,\n
\"min\": 166,\n        \"max\": 282,\n        \"num_unique_values\":
5,\n        \"samples\": [\n          282\n        ],\n
```

```
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```

```
(227,)
```

```
209    1
183    0
165    0
41     0
291    0
Name: AHD, dtype: int64
```

```
(76,)
```

```
<bound method NDFrame.head of 288    0
223    1
264    1
96     1
219    0
      ..
55     1
79     1
217    0
287    0
188    1
Name: AHD, Length: 76, dtype: int64>
```

```
x_train['ChestPain'].unique()
```

```
array(['asymptomatic', 'typical', 'nontypical', 'nonanginal'],
      dtype=object)
```

```
from sklearn.preprocessing import OrdinalEncoder

ordinal_encoder = OrdinalEncoder(categories=[['typical',
'asymptomatic', 'nonanginal', 'nontypical']])
x_train['ChestPain'] =
ordinal_encoder.fit_transform(x_train[['ChestPain']])
x_test['ChestPain'] = ordinal_encoder.transform(x_test[['ChestPain']])
print(x_train.head())
print(x_test.head())
```

```
     Age  Sex  ChestPain  RestBP  Chol
209   62    0        1.0     150   244
183   59    1        0.0     178   270
165   57    1        1.0     132   207
41    40    1        0.0     140   199
291   55    0        3.0     132   342
     Age  Sex  ChestPain  RestBP  Chol
288   56    1        3.0     130   221
223   53    1        1.0     123   282
```

```
264   61    1        1.0     138   166
96    59    1        1.0     110   239
219   59    1        1.0     138   271
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
x_train[['Age', 'RestBP', 'Chol']] =
scaler.fit_transform(x_train[['Age', 'RestBP', 'Chol']])
x_test[['Age', 'RestBP', 'Chol']] = scaler.transform(x_test[['Age',
'RestBP', 'Chol']])

print(x_train.head())
print(x_test.head())
```

```
          Age  Sex  ChestPain    RestBP      Chol
209  0.834084    0        1.0  1.014683 -0.025148
183  0.508203    1        0.0  2.627015  0.469594
165  0.290948    1        1.0 -0.021816 -0.729203
41  -1.555713    1        0.0  0.438850 -0.881431
291  0.073694    0        3.0 -0.021816  1.839647
          Age  Sex  ChestPain    RestBP      Chol
288  0.182321    1        3.0 -0.136982 -0.462804
223 -0.143560    1        1.0 -0.540065  0.697936
264  0.725457    1        1.0  0.323684 -1.509372
96   0.508203    1        1.0 -1.288648 -0.120290
219  0.508203    1        1.0  0.323684  0.488622
```

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

```
LogisticRegression()
```

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

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

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

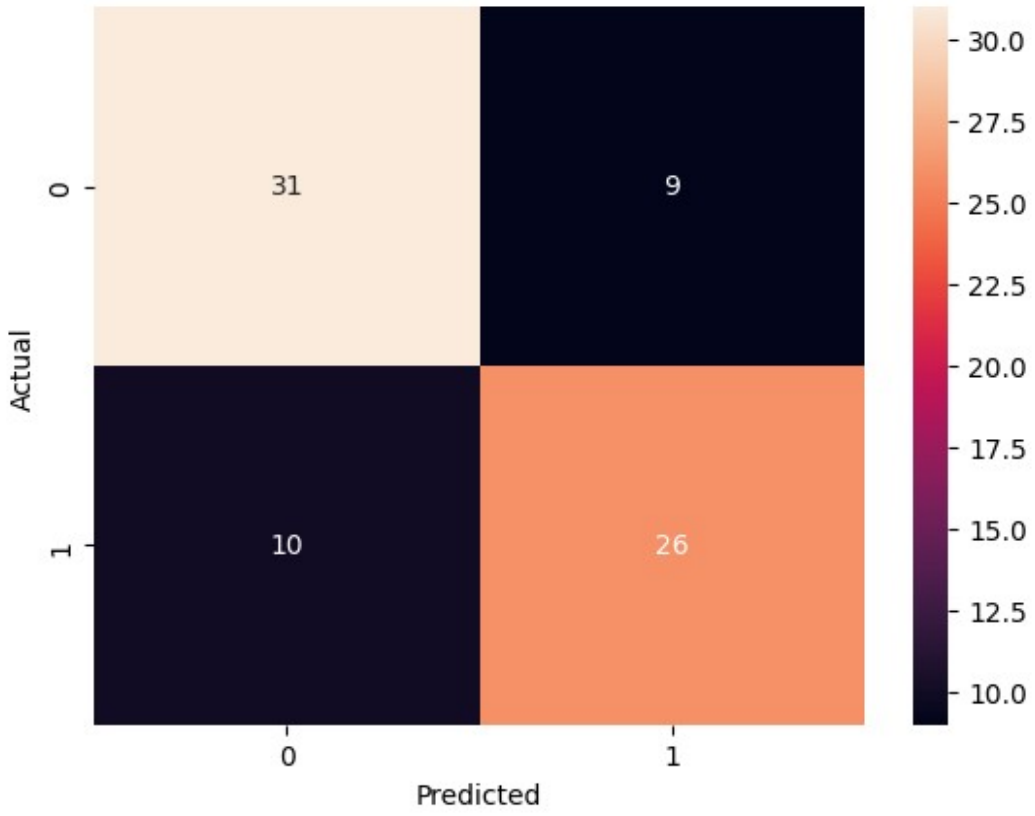

```
{"summary":"{\n  \"name\": \"pd\",\n  \"rows\": 76,\n  \"fields\": [\n
{\n      \"column\": \"y_test\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
```

```
[\n          1,\n          0\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"}
```

```
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, confusion_matrix, classification_report
print('Accuracy Score: ', accuracy_score(y_test, y_pred))
print('Precision Score: ', precision_score(y_test, y_pred))
print('Recall Score: ', recall_score(y_test, y_pred))
print('F1 Score: ', f1_score(y_test, y_pred))
print('Confusion Matrix:\n ', confusion_matrix(y_test, y_pred))
print('Classification Report:\n ', classification_report(y_test,
y_pred))
```

```
Accuracy Score:  0.75
Precision Score:  0.7428571428571429
Recall Score:  0.7222222222222222
F1 Score:  0.7323943661971831
Confusion Matrix:
  [[31  9]
 [10 26]]
Classification Report:
                precision    recall  f1-score   support

           0       0.76      0.78      0.77        40
           1       0.74      0.72      0.73        36

    accuracy                           0.75        76
   macro avg       0.75      0.75      0.75        76
weighted avg       0.75      0.75      0.75        76
```

```
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```


31
9
10
26
0
1
Actual
0
1
Predicted
30.0
27.5
25.0
22.5
20.0
17.5
15.0
12.5
10.0

Muhammad Umer Adeeb

Question 1: Mall_Customer Dataset

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

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


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 200,\n  \"fields\": [\
n    {\n      \"column\": \"CustomerID\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 57,\n        \"min\": 1,\n
\"max\": 200,\n        \"num_unique_values\": 200,\n
\"samples\": [\n          96,\n          16,\n          31\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Gender\",\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\": 13,\n        \"min\": 18,\n
\"max\": 70,\n        \"num_unique_values\": 51,\n        \"samples\":
[\n          55,\n          26\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Annual Income (k$)\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 26,\n        \"min\": 15,\n
\"max\": 137,\n        \"num_unique_values\": 64,\n
\"samples\": [\n          87,\n          101\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Spending Score (1-100)\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
25,\n        \"min\": 1,\n        \"max\": 99,\n
\"num_unique_values\": 84,\n        \"samples\": [\n          83,\n
39\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


Objective: Group retail store customers based on their purchase history.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
```

```
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

```
df.shape
```

```
(200, 5)
```

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

```
CustomerID                  0
Gender                      0
Age                         0
Annual Income (k$)          0
Spending Score (1-100)      0
dtype: int64
```

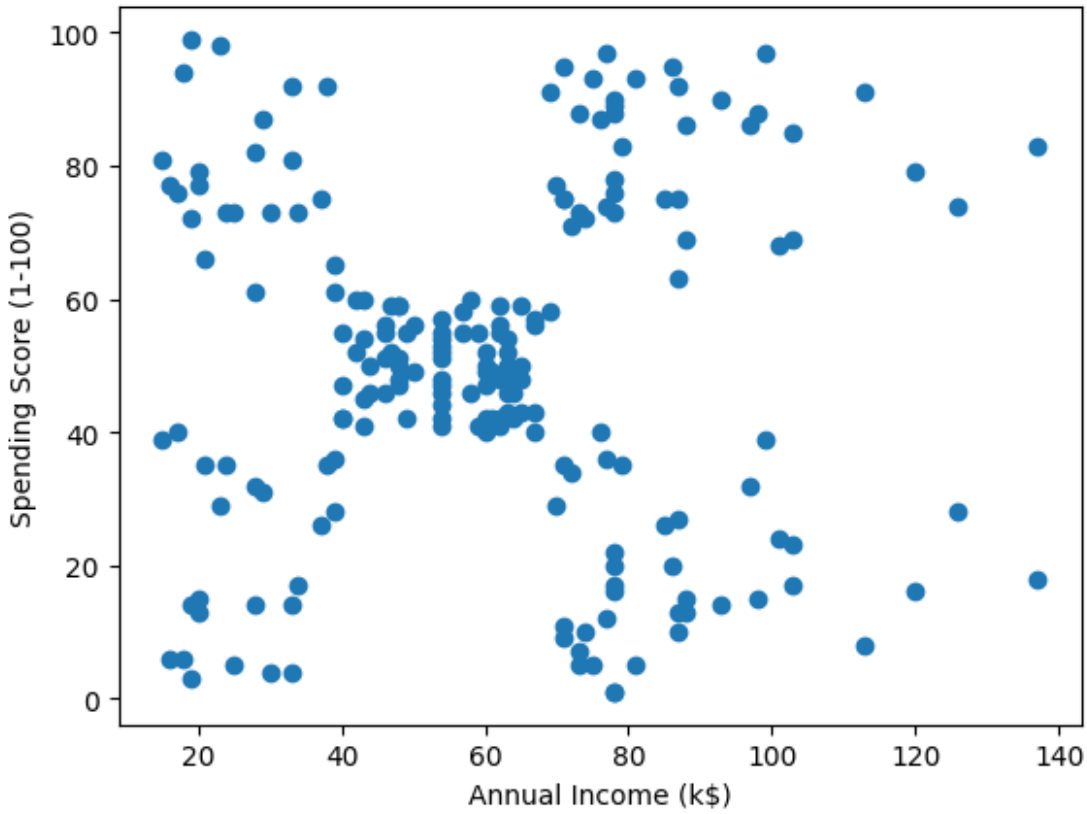
```
duplicate_rows_df = df[df.duplicated()]
print("Number of duplicate rows: ", duplicate_rows_df.shape)
duplicate_rows_df
```

```
Number of duplicate rows:  (0, 5)
```


```
{"repr_error":"Out of range float values are not JSON compliant:
nan","type":"dataframe","variable_name":"duplicate_rows_df"}
```


```
X = df[['Annual Income (k$)', 'Spending Score (1-100)']]
```

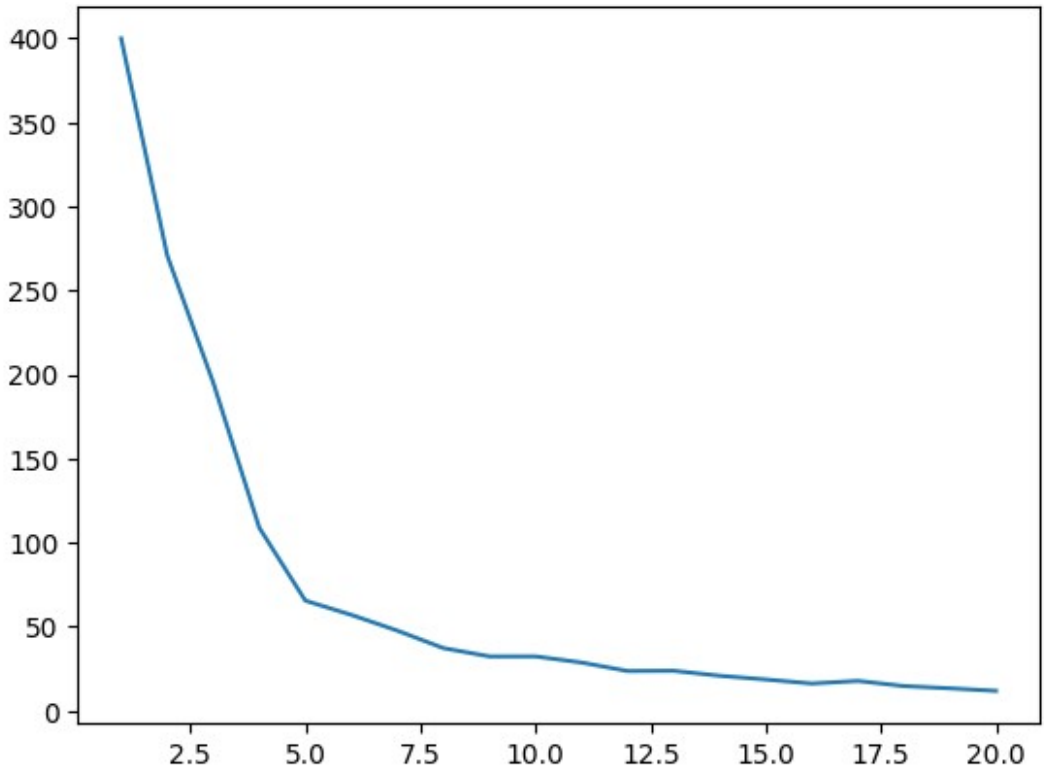
```
plt.scatter(X['Annual Income (k$)'], X['Spending Score (1-100)'])
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.show()
```




```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

from sklearn.cluster import KMeans
wcss = []
for i in range(1, 21):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

wcss
```

```
[399.99999999999994,
 270.89235946739063,
 195.2466301907915,
 108.92131661364358,
 65.57885579985046,
 57.11147724296594,
 47.710583761307916,
 37.31912287833882,
 32.39226763033118,
 32.40246298115112,
 28.751291042159014,
 23.710344944514176,
```

```
 23.848393399486483,
 20.877880368245425,
 18.681695442209257,
 16.27860877692049,
 17.83632250802142,
 14.763920183001115,
 13.42240191419025,
 11.900757680608644]

plt.plot(range(1,21), wcss)

[<matplotlib.lines.Line2D at 0x7f7811484950>]
```




```
km = KMeans(n_clusters=5)
y_means = km.fit_predict(X_scaled)

y_means

array([3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3,
1,
       3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3,
0,
       3, 1, 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, 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, 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, 2, 4, 2, 0, 2, 4, 2, 4,
2,
       0, 2, 4, 2, 4, 2, 4, 2, 4, 2, 0, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4,
2,
       4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4,
2,
       4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4,
2,
       4, 2], dtype=int32)
```

```
X_scaled[y_means ==0]
```

```
array([[-0.82293289,  0.41927286],
       [-0.78476346,  0.18634349],
       [-0.78476346, -0.12422899],
       [-0.78476346, -0.3183368 ],
       [-0.78476346, -0.3183368 ],
       [-0.70842461,  0.06987881],
       [-0.70842461,  0.38045129],
       [-0.67025518,  0.14752193],
       [-0.67025518,  0.38045129],
       [-0.67025518, -0.20187212],
       [-0.67025518, -0.35715836],
       [-0.63208575, -0.00776431],
       [-0.63208575, -0.16305055],
       [-0.55574689,  0.03105725],
       [-0.55574689, -0.16305055],
       [-0.55574689,  0.22516505],
       [-0.55574689,  0.18634349],
       [-0.51757746,  0.06987881],
       [-0.51757746,  0.34162973],
       [-0.47940803,  0.03105725],
       [-0.47940803,  0.34162973],
       [-0.47940803, -0.00776431],
       [-0.47940803, -0.08540743],
       [-0.47940803,  0.34162973],
       [-0.47940803, -0.12422899],
       [-0.4412386 ,  0.18634349],
       [-0.4412386 , -0.3183368 ],
       [-0.40306917, -0.04658587],
       [-0.40306917,  0.22516505],
       [-0.25039146, -0.12422899],
       [-0.25039146,  0.14752193],
       [-0.25039146,  0.10870037],
       [-0.25039146, -0.08540743],
       [-0.25039146,  0.06987881],
       [-0.25039146, -0.3183368 ],
```

```
       [-0.25039146,  0.03105725],
       [-0.25039146,  0.18634349],
       [-0.25039146, -0.35715836],
       [-0.25039146, -0.24069368],
       [-0.25039146,  0.26398661],
       [-0.25039146, -0.16305055],
       [-0.13588317,  0.30280817],
       [-0.13588317,  0.18634349],
       [-0.09771374,  0.38045129],
       [-0.09771374, -0.16305055],
       [-0.05954431,  0.18634349],
       [-0.05954431, -0.35715836],
       [-0.02137488, -0.04658587],
       [-0.02137488, -0.39597992],
       [-0.02137488, -0.3183368 ],
       [-0.02137488,  0.06987881],
       [-0.02137488, -0.12422899],
       [-0.02137488, -0.00776431],
       [ 0.01679455, -0.3183368 ],
       [ 0.01679455, -0.04658587],
       [ 0.05496398, -0.35715836],
       [ 0.05496398, -0.08540743],
       [ 0.05496398,  0.34162973],
       [ 0.05496398,  0.18634349],
       [ 0.05496398,  0.22516505],
       [ 0.05496398, -0.3183368 ],
       [ 0.09313341, -0.00776431],
       [ 0.09313341, -0.16305055],
       [ 0.09313341, -0.27951524],
       [ 0.09313341, -0.08540743],
       [ 0.09313341,  0.06987881],
       [ 0.09313341,  0.14752193],
       [ 0.13130284, -0.3183368 ],
       [ 0.13130284, -0.16305055],
       [ 0.16947227, -0.08540743],
       [ 0.16947227, -0.00776431],
       [ 0.16947227, -0.27951524],
       [ 0.16947227,  0.34162973],
       [ 0.24581112, -0.27951524],
       [ 0.24581112,  0.26398661],
       [ 0.24581112,  0.22516505],
       [ 0.24581112, -0.39597992],
       [ 0.32214998,  0.30280817],
       [ 0.39848884, -0.59008772],
       [ 0.43665827, -0.62890928],
       [ 0.58933599, -0.39597992]])

plt.scatter(X_scaled[y_means==0,0] , X_scaled[y_means==0,1],
color='blue')
plt.scatter(X_scaled[y_means==1,0], X_scaled[y_means==1,1],
```

```
color='red')
plt.scatter(X_scaled[y_means==2,0], X_scaled[y_means==2,1],
color='green')
plt.scatter(X_scaled[y_means==3,0], X_scaled[y_means==3,1],
color='yellow')
plt.scatter(X_scaled[y_means==4,0], X_scaled[y_means==4,1],
color='black')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.show()
```