KNN

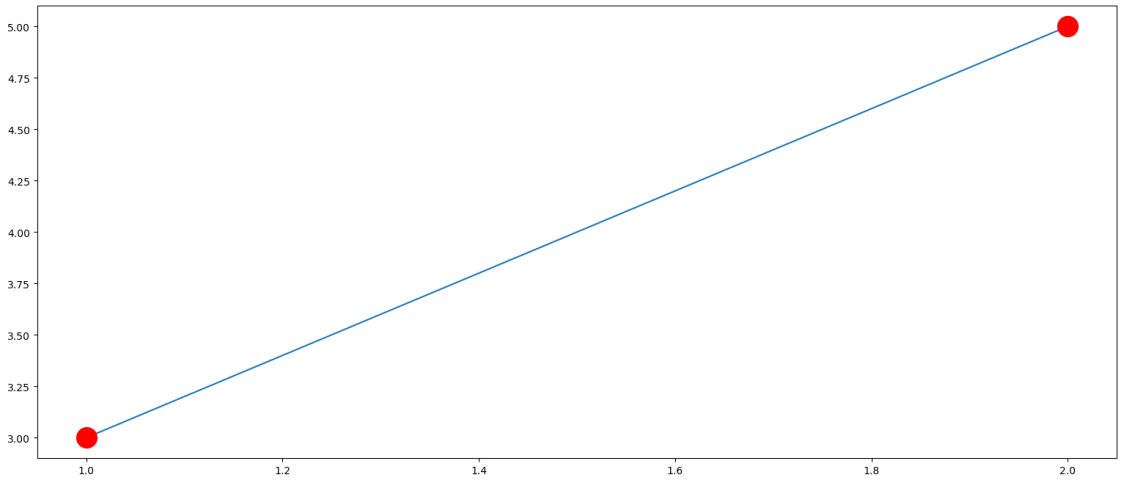
November 18, 2024

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

[3]:
```
plt.rcParams['figure.figsize'] = [19, 8]
```

[5]:
```
import warnings
warnings.filterwarnings('ignore')
```

[7]:
```
x = np.array([1, 2])
y = np.array([3, 5])

plt.plot(x, y, marker = 'o', ms=20, mec='r', mfc='r')
plt.show()
```



[9]:
```
data = {
    'Height': [158, 158, 158, 160, 160, 163, 163, 160, 163, 165, 165, 165, 168,␣
 ↪168, 168, 170, 170, 170],
    'Weight': [58, 59, 63, 59, 60, 60, 61, 64, 64, 61, 62, 65, 62, 63, 66, 63,␣
 ↪64, 68],
```

```
    'ShirtSize': ['M', 'M',␣
 ↪'M','M','M','M','M','L','L','L','L','L','L','L','L','L','L','L',]
}

df = pd.DataFrame(data)

df.head()
```

[9]:

	Height	Weight	ShirtSize
0	158	58	M
1	158	59	M
2	158	63	M
3	160	59	M
4	160	60	M

[11]:
```
new_height = 161
new_weight = 61
```

[13]:
```
df['Distance'] = np.sqrt((df['Height'] - new_height)**2 + (df['Weight'] -␣
 ↪new_weight)**2)
```

[15]:
```
df.head()
```

[15]:

	Height	Weight	ShirtSize	Distance
0	158	58	M	4.242641
1	158	59	M	3.605551
2	158	63	M	3.605551
3	160	59	M	2.236068
4	160	60	M	1.414214

[17]:
```
k_value = 5
```

[19]:
```
df.sort_values(by='Distance', ascending=True).head(k_value)
```

[19]:

	Height	Weight	ShirtSize	Distance
4	160	60	M	1.414214
6	163	61	M	2.000000
3	160	59	M	2.236068
5	163	60	M	2.236068
7	160	64	L	3.162278

[23]:
```
diabetes_df = pd.read_csv('diabetes.csv')
```

[25]:
```
diabetes_df.shape
```

[25]: (768, 9)

[27]:
```
diabetes_df.head()
```

[27]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

[29]:
```
diabetes_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

[31]:
```
for column in diabetes_df.columns:
    print(f'{column}: {diabetes_df.loc[diabetes_df[column] == 0, column].
 ↪count()}')
```

```
Pregnancies: 111
Glucose: 5
BloodPressure: 35
SkinThickness: 227
Insulin: 374
BMI: 11
DiabetesPedigreeFunction: 0
Age: 0
Outcome: 500
```

[33]:
```
columns = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']

for column in columns:
    diabetes_df[column] = diabetes_df[column].replace(0, np.nan)
```

[35]:
```
diabetes_df.info()
```

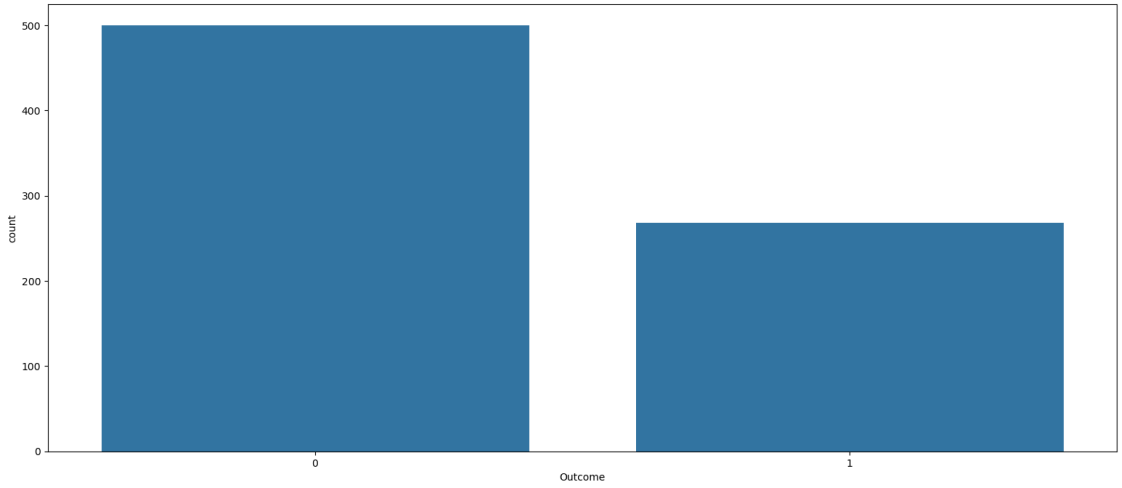
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   763 non-null    float64
 2   BloodPressure             733 non-null    float64
 3   SkinThickness             541 non-null    float64
 4   Insulin                   394 non-null    float64
 5   BMI                       757 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(6), int64(3)
memory usage: 54.1 KB
```

[37]:
```
columns = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']

for column in columns:
    diabetes_df[column].fillna(diabetes_df[column].median(), inplace=True)
```

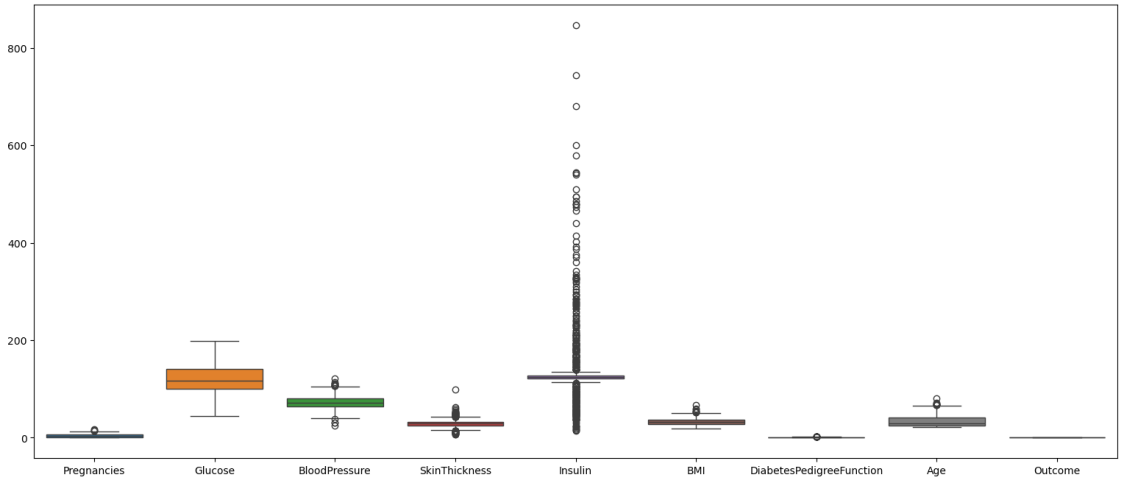
[39]:
```
diabetes_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    float64
 2   BloodPressure             768 non-null    float64
 3   SkinThickness             768 non-null    float64
 4   Insulin                   768 non-null    float64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(6), int64(3)
memory usage: 54.1 KB
```

[41]:
```
sns.countplot(data=diabetes_df, x='Outcome')

plt.show()
```



[43]:
```
sns.boxplot(diabetes_df)

plt.show()
```



[45]:
```
# obtain the first quartile
Q1 = diabetes_df.quantile(0.25)

# obtain the third quartile
Q3 = diabetes_df.quantile(0.75)
```

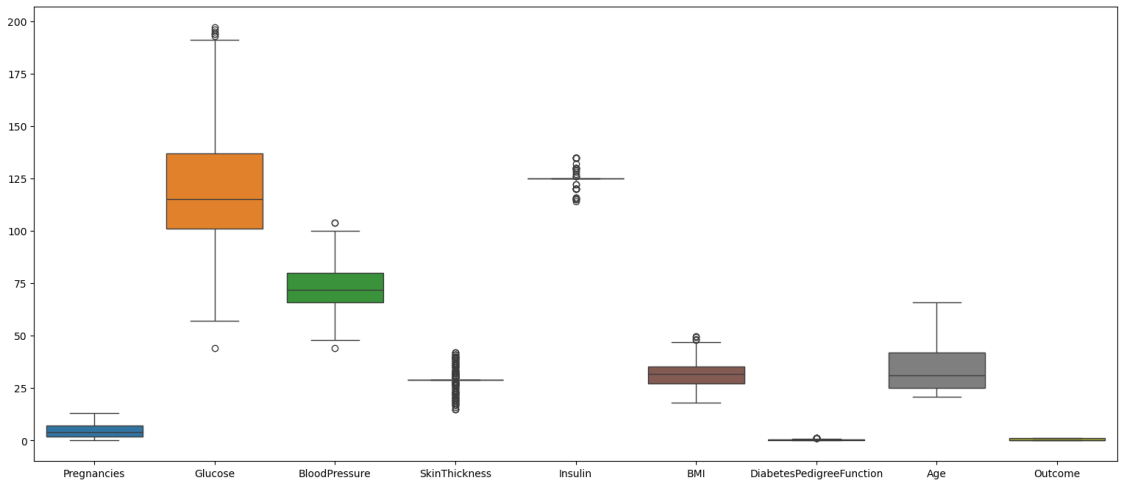
```
# obtain the IQR
IQR = Q3 - Q1

# print the IQR
print(IQR)
```

```
Pregnancies                 5.0000
Glucose                    40.5000
BloodPressure              16.0000
SkinThickness               7.0000
Insulin                     5.7500
BMI                         9.1000
DiabetesPedigreeFunction    0.3825
Age                        17.0000
Outcome                     1.0000
dtype: float64
```

[47]:
```
ul = Q3 + 1.5 * IQR

ll = Q1 - 1.5 * IQR
```

[49]:
```
diabetes_df = diabetes_df[~((diabetes_df < ll) |(diabetes_df > ul)).any(axis=1)]
```

[51]:
```
sns.boxplot(diabetes_df)

plt.show()
```



[53]:
```
diabetes_df.shape
```

[53]: (375, 9)

[55]:
```
X = diabetes_df.loc[:, :'Age'].values

y = diabetes_df.loc[:, 'Outcome'].values
```

[57]:
```
from sklearn.preprocessing import StandardScaler
```

[59]:
```
scaler = StandardScaler()
```

[61]:
```
X = scaler.fit_transform(X)
```

[63]:
```
from sklearn.model_selection import train_test_split
```

[65]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=2)
```

[67]:
```
y_test.shape
```

[67]:
```
(75,)
```

[69]:
```
from sklearn.neighbors import KNeighborsClassifier
```

[71]:
```
for k in range(1, 20):
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(X_train, y_train)
    accuracy = model.score(X_test, y_test)
    print(f'K: {k}\tAccuracy:{accuracy:.2%}')
```

```
K: 1    Accuracy:66.67%
K: 2    Accuracy:64.00%
K: 3    Accuracy:70.67%
K: 4    Accuracy:73.33%
K: 5    Accuracy:77.33%
K: 6    Accuracy:76.00%
K: 7    Accuracy:74.67%
K: 8    Accuracy:70.67%
K: 9    Accuracy:74.67%
K: 10   Accuracy:74.67%
K: 11   Accuracy:74.67%
K: 12   Accuracy:76.00%
K: 13   Accuracy:73.33%
K: 14   Accuracy:74.67%
K: 15   Accuracy:76.00%
K: 16   Accuracy:73.33%
K: 17   Accuracy:73.33%
K: 18   Accuracy:70.67%
K: 19   Accuracy:77.33%
```

[73]:
```
model = KNeighborsClassifier(n_neighbors=5)
```

[75]:
```
model.fit(X_train, y_train)
```

[75]:
```
KNeighborsClassifier()
```

[77]:
```
model.score(X_train, y_train)
```

[77]:
```
0.8033333333333333
```

[79]:
```
model.score(X_test, y_test)
```

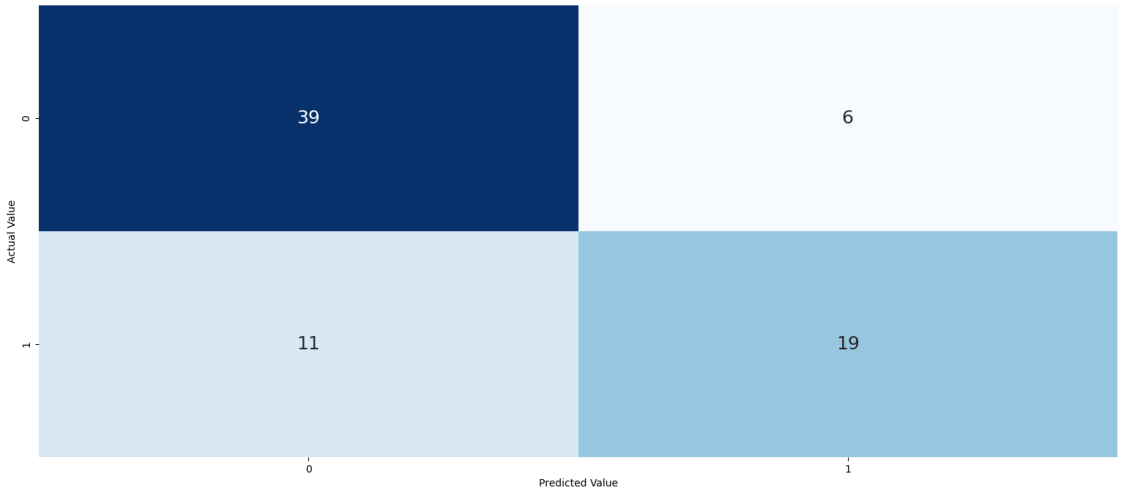
[79]:
```
0.7733333333333333
```

[81]:
```
y_predict = model.predict(X_test)
```

[83]:
```
from sklearn.metrics import confusion_matrix
```

[85]:
```
cm = confusion_matrix(y_test, y_predict)
```

[87]:
```
sns.heatmap(cm, annot=True, cmap='Blues', cbar=False, annot_kws={"fontsize":18})

plt.xlabel("Predicted Value")

plt.ylabel("Actual Value")

plt.show()
```



[89]:
```
from sklearn.metrics import classification_report
```

[91]:
```
print(classification_report(y_test, y_predict))
```

	precision	recall	f1-score	support
0	0.78	0.87	0.82	45
1	0.76	0.63	0.69	30
accuracy			0.77	75
macro avg	0.77	0.75	0.76	75
weighted avg	0.77	0.77	0.77	75

[ ]: