CMPE428 Assignment 4

Building Nearest Neighbour Classifiers by Çağıl Peköz

Imports

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

from math import sqrt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score
```

Import CSV

```
df = pd.read_csv('stdData 1.csv')
```

```
df.head()
```

	Label	V1	V2	V3	V4	V5	V6	V7
0	positive	1.221400	128.101200	80.035036	35.431417	180.956968	42.944951	1.320305
1	negative	2.609743	85.891549	58.543681	14.454311	52.545356	33.426070	-0.786571
2	negative	2.682163	99.782456	68.000884	26.339627	71.578043	37.542894	0.534953
3	negative	3.196969	115.189168	65.307845	-0.539337	0.269863	20.857287	0.562433
4	positive	4.790932	144.487763	80.800220	18.937774	-0.033570	31.346055	0.789162

Task

Split Dataset with Equal Positives and Negatives

We can see the amount of negatives and positives with this command

```
df.Label.value_counts()
```

```
negative    195
positive    105
Name: Label, dtype: int64
```

Replacing Categorical with Binary

```
df["Label"] = df["Label"].replace({"positive":1,"negative":0})
```

```
df.head()
```

	Label	V1	V2	V3	V4	V5	V6	V7
0	1	1.221400	128.101200	80.035036	35.431417	180.956968	42.944951	1.320305
1	0	2.609743	85.891549	58.543681	14.454311	52.545356	33.426070	-0.786571
2	0	2.682163	99.782456	68.000884	26.339627	71.578043	37.542894	0.534953
3	0	3.196969	115.189168	65.307845	-0.539337	0.269863	20.857287	0.562433
4	1	4.790932	144.487763	80.800220	18.937774	-0.033570	31.346055	0.789162

Splitting Dataset Into 2, with Equal Amounts of Negative and Positive

We first get our X and Y values.

```
X = df.drop(["Label"], axis = 1)
y = df["Label"]
```

We use stratify on y and split our dataframe into 2, so that we get equal distribution of negatives and positives.

```
X_train, X_test = train_test_split(df, test_size = 0.5, stratify = y)
```

```
y_train.value_counts()
```

```
0    98
1    52
Name: Label, dtype: int64
```

```
y_test.value_counts()
```

```
0    97
1    53
Name: Label, dtype: int64
```

We can see that numbers of 0 and 1 is equal across test and train.

kNN Classifier and Scores

Here I have built the model and for test purposes I tried k=1.

```
neigh = KNeighborsClassifier(n_neighbors=1)
neigh.fit(X_train,y_train)
y_pred = neigh.predict(X_test)

scores = {}
scores['K Value'] = "1"
scores['Accuracy'] = accuracy_score(y_test, y_pred)
scores['F1'] = f1_score(y_test, y_pred)
scores['Recall'] = recall_score(y_test, y_pred)
scores['Precision'] = precision_score(y_test, y_pred)

scores_df = pd.DataFrame.from_dict(scores, orient='index')
scores_df = scores_df.transpose()
scores_df
```

	K Value	Accuracy	F1	Recall	Precision
0	1	0.586667	0.354167	0.320755	0.395349

Testing Different K Values

Here I will try k=2.

```
neigh = KNeighborsClassifier(n_neighbors=2)
neigh.fit(X_train,y_train)
y_pred = neigh.predict(X_test)

scores = {}
scores['K Value'] = "2"
scores['Accuracy'] = accuracy_score(y_test, y_pred)
scores['F1'] = f1_score(y_test, y_pred)
scores['Recall'] = recall_score(y_test, y_pred)
scores['Precision'] = precision_score(y_test, y_pred)

scores_df = scores_df.append(scores, ignore_index=True)
scores_df
```

	K Value	Accuracy	F1	Recall	Precision
0	1	0.586667	0.354167	0.320755	0.395349
1	2	0.586667	0.0882353	0.0566038	0.2

Here I will try k=3.

```
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(X_train,y_train)
y_pred = neigh.predict(X_test)

scores = {}
scores['K Value'] = "3"
scores['Accuracy'] = accuracy_score(y_test, y_pred)
scores['F1'] = f1_score(y_test, y_pred)
scores['Recall'] = recall_score(y_test, y_pred)
scores['Precision'] = precision_score(y_test, y_pred)

scores_df = scores_df.append(scores, ignore_index=True)
scores_df
```

	K Value	Accuracy	F1	Recall	Precision
0	1	0.586667	0.354167	0.320755	0.395349
1	2	0.586667	0.0882353	0.0566038	0.2
2	3	0.566667	0.235294	0.188679	0.3125

Trying Out a Few Formulas I Found Online

According to my research, as a general rule of thumb, k-value is determined by k = sqrt(N)/2 or k = sqrt(N) formula. I will use both of these formulas and check the scores.

```
k1 = sqrt(len(X_train))/2
round(k1)
```

```
6
```

```
k2 = sqrt(len(X_train))
round(k2)
```

```
12
```

Now I will try k=6.

```
neigh = KNeighborsClassifier(n_neighbors=6)
neigh.fit(X_train,y_train)
y_pred = neigh.predict(X_test)

scores = {}
scores['K Value'] = "6"
scores['Accuracy'] = accuracy_score(y_test, y_pred)
scores['F1'] = f1_score(y_test, y_pred)
scores['Recall'] = recall_score(y_test, y_pred)
```

```
scores['Precision'] = precision_score(y_test, y_pred)

scores_df = scores_df.append(scores, ignore_index=True)
scores_df
```

	K Value	Accuracy	F1	Recall	Precision
0	1	0.586667	0.354167	0.320755	0.395349
1	2	0.586667	0.0882353	0.0566038	0.2
2	3	0.566667	0.235294	0.188679	0.3125
3	6	0.613333	0.236842	0.169811	0.391304

Lastly, I will try k=12.

```
neigh = KNeighborsClassifier(n_neighbors=12)
neigh.fit(X_train,y_train)
y_pred = neigh.predict(X_test)

scores = {}
scores['K Value'] = "12"
scores['Accuracy'] = accuracy_score(y_test, y_pred)
scores['F1'] = f1_score(y_test, y_pred)
scores['Recall'] = recall_score(y_test, y_pred)
scores['Precision'] = precision_score(y_test, y_pred)

scores_df = scores_df.append(scores, ignore_index=True)
scores_df
```

	K Value	Accuracy	F1	Recall	Precision
0	1	0.586667	0.354167	0.320755	0.395349
1	2	0.586667	0.0882353	0.0566038	0.2
2	3	0.566667	0.235294	0.188679	0.3125
3	6	0.613333	0.236842	0.169811	0.391304
4	12	0.653333	0.212121	0.132075	0.538462

In here, we can see that our Precision and Accuracy is all time high in the k-value 12, however our F1 and Recall went down. In k-value 6, our F1 and Recall numbers were higher.