ASSIGNMENT 4 MACHINE LEARNING

SYABAB ALHAQQI BIN JAAFAR 2211117

Problem Statement:

Cardiovascular diseases are a leading cause of mortality worldwide. Predicting the risk of heart problems can be crucial for early intervention and preventive measures. In this assignment you are tasked with developing a predictive model using computational intelligence techniques to assess the likelihood of an individual experiencing heart problems.

1) Data collection:

- Handle Missing Values: Use median for numerical data and mode for categorical.
- Normalize: Use MinMaxScaler for scaling numerical features.
- Standardize: Use StandardScaler for standardizing numerical features.
- Encode Categorical Features: Use LabelEncoder or OneHotEncoder as appropriate.
- Train-Test Split: Split data into training and testing sets using train_test_split().

Most significant features:

- Age
- Cholesterol
- Blood pressure
- History

Least significant features:

- Income
- Country

2) Logistic Regression vs KNN

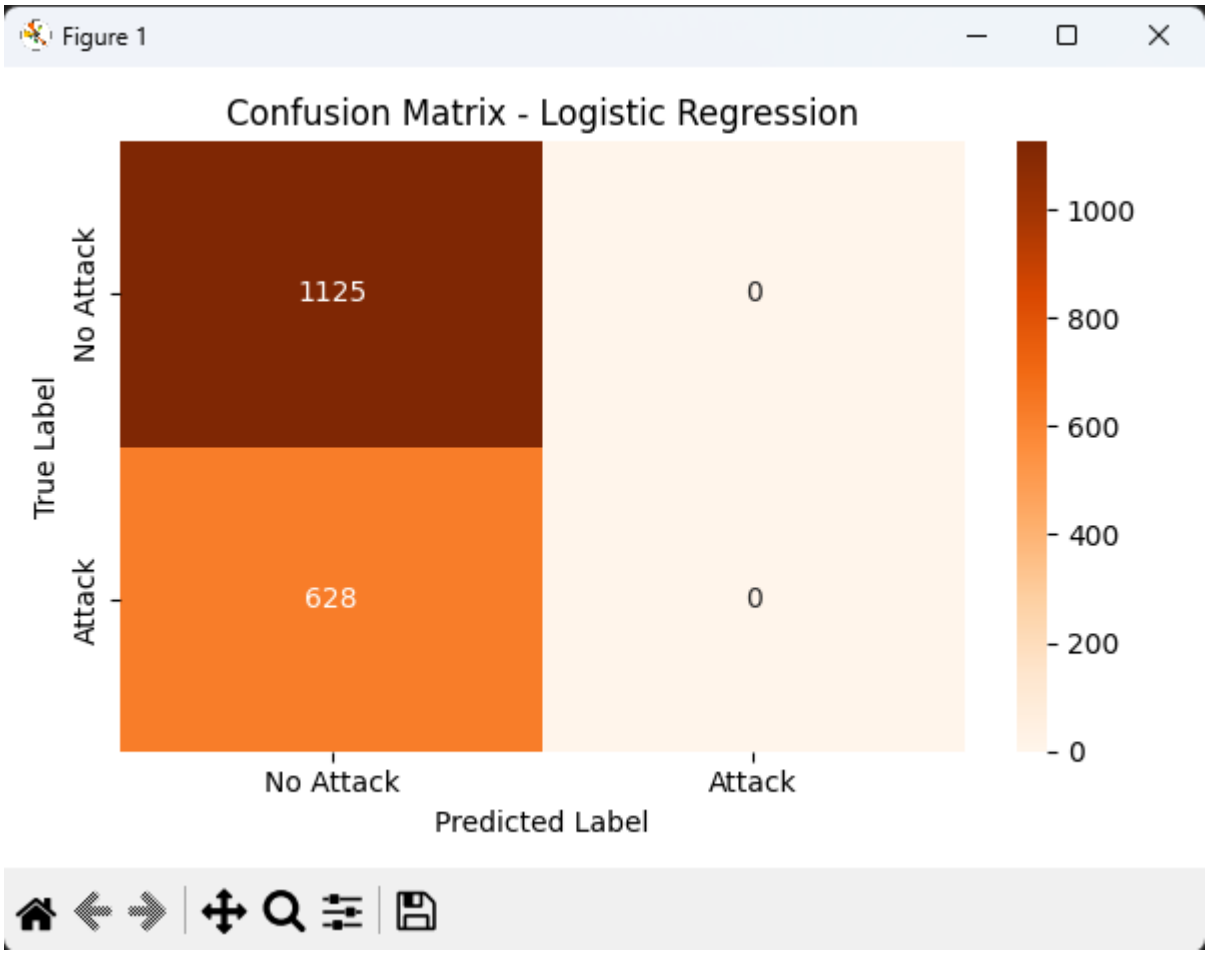
a) Logistic Regression

Coding:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Load dataset
file_path = "heart_attack_prediction_dataset_Original.xlsx"
df = pd.read_excel(file_path)

# Preprocess Blood Pressure: split and average
bp_split = df['Blood Pressure'].str.split('/', expand=True)
df['Systolic BP'] = pd.to_numeric(bp_split[0], errors='coerce')
df['Diastolic BP'] = pd.to_numeric(bp_split[1], errors='coerce')
df['BP_Avg'] = (df['Systolic BP'] + df['Diastolic BP']) / 2

# Select required features
features = ['Age', 'Cholesterol', 'BP_Avg', 'Previous Heart Problems']
df_selected = df[features + ['Heart Attack Risk']].dropna()

# Ensure numeric data
df_selected['Cholesterol'] = pd.to_numeric(df_selected['Cholesterol'], errors='coerce')
df_selected['Previous Heart Problems'] = pd.to_numeric(df_selected['Previous Heart Problems'], errors='coerce')
df_selected = df_selected.dropna()
```

```
# Define features and target
X = df_selected[features]
y = df_selected['Heart Attack Risk'].astype(int)

# Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=24)

# Logistic Regression model
log_model = LogisticRegression()
log_model.fit(X_train, y_train)
y_pred = log_model.predict(X_test)

# Evaluation
print("\n----- Logistic Regression Results -----")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))

# Confusion Matrix Visualization
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Oranges',
        xticklabels=['No Attack', 'Attack'],
        yticklabels=['No Attack', 'Attack'])
plt.title("Confusion Matrix - Logistic Regression")
plt.xlabel("Predicted Label")
```

```
plt.ylabel("True Label")
plt.tight_layout()
plt.show()
```



b) KNN

Coding:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix

# Step 1: Load the dataset
file_path = "heart_attack_prediction_dataset_Original.xlsx"
df = pd.read_excel(file_path)  # Default sheet

# Step 2: Data Preprocessing

# Split 'Blood Pressure' into systolic and diastolic
bp_split = df["Blood Pressure"].str.split("/", expand=True)
df["Systolic BP"] = pd.to_numeric(bp_split[0], errors='coerce')
df["Diastolic BP"] = pd.to_numeric(bp_split[1], errors='coerce')

# Select only relevant features
selected_features = ["Age", "Cholesterol", "Systolic BP", "Diastolic BP", "Heart Rate",
"Heart Attack Risk"]
df = df[selected_features]

# Drop missing values
df.dropna(inplace=True)

# Separate features and target
```

```
X = df.drop(columns=["Heart Attack Risk"])
y = df["Heart Attack Risk"]

# Standardize features
sc = StandardScaler()
X_scaled = sc.fit_transform(X)

# Step 3: Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)

# Step 4: KNN Classifier
knn_model = KNeighborsClassifier(n_neighbors=6)
knn_model.fit(X_train, y_train)
y_pred = knn_model.predict(X_test)

# Step 5: Evaluation
print("\n----- KNN Classifier Results -----")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)

# Step 6: Seaborn Heatmap for Confusion Matrix
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
        xticklabels=['No Attack', 'Attack'],
        yticklabels=['No Attack', 'Attack'])
plt.title("Confusion Matrix - KNN Classifier")
plt.xlabel("Predicted Label")
```

```
plt.ylabel("True Label")
plt.tight_layout()
plt.show()

# Optional: Accuracy vs K-Value Plot
accuracies = []
k_range = range(1, 21)
for k in k_range:
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(X_train, y_train)
    preds = model.predict(X_test)
    accuracies.append(accuracy_score(y_test, preds))

plt.figure(figsize=(8, 4))
plt.plot(k_range, accuracies, marker='o')
plt.title('KNN Accuracy for Different k Values')
plt.xlabel('Number of Neighbors (k)')
plt.ylabel('Accuracy')
plt.grid(True)
plt.show()
```

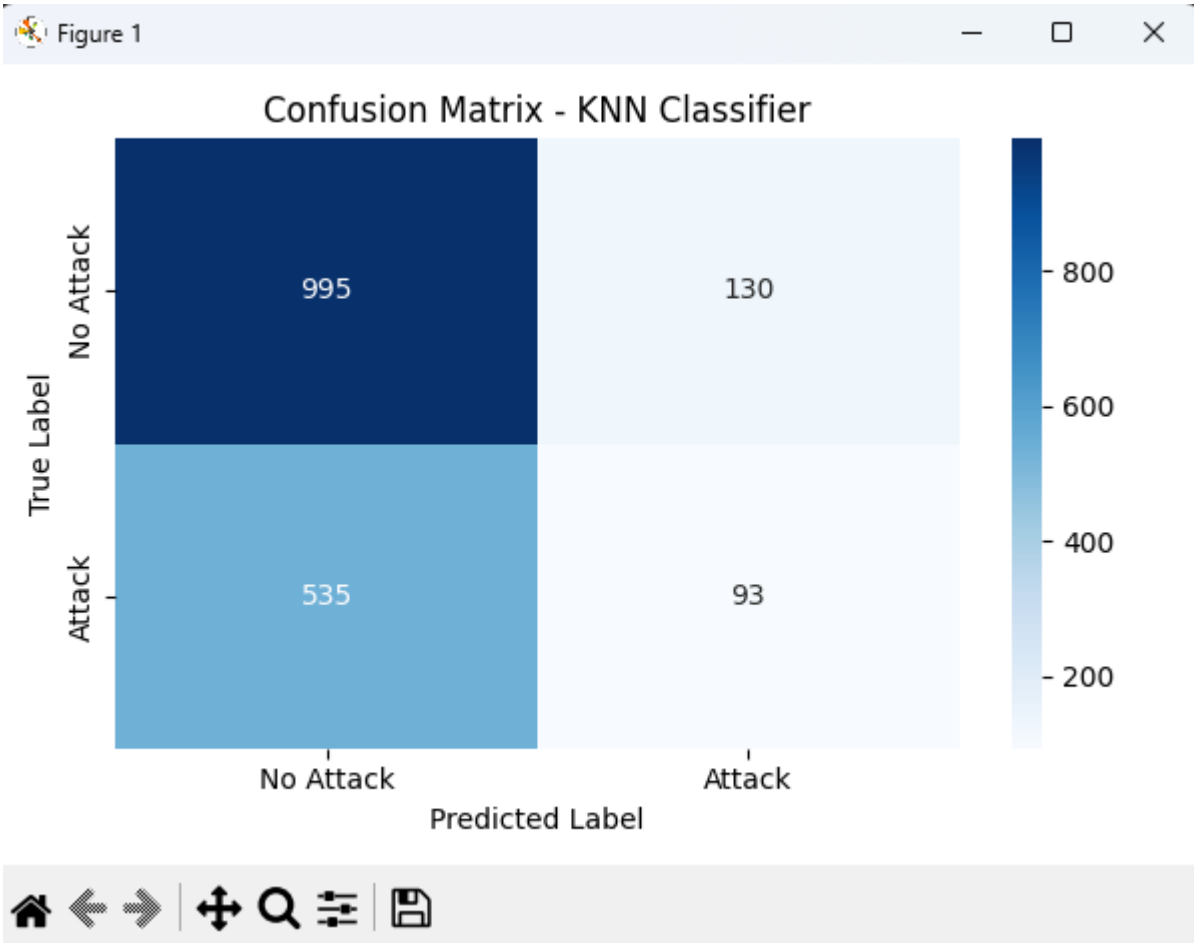


Figure 1
Confusion Matrix - KNN Classifier
995
130
535
93
No Attack
Attack
True Label
No Attack
Attack
Predicted Label
800
600
400
200

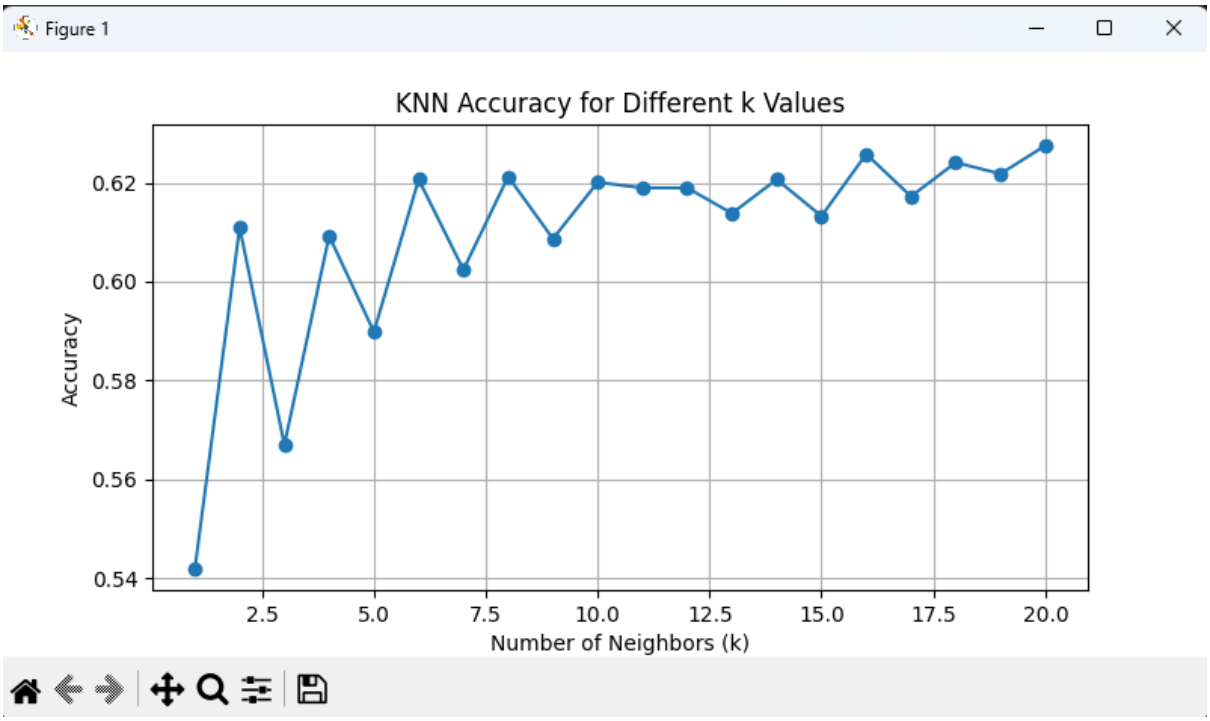


Figure 1
KNN Accuracy for Different k Values
Accuracy
0.62
0.60
0.58
0.56
0.54
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Number of Neighbors (k)

3) Conclusion

If we want interpretability and speed, Logistic Regression is a better choice but if we want accuracy and non-linear patterns matter more, KNN is better. Therefore, for this type of data, it is preferred using KNN for predicting the heart attack with a higher accuracy.