*1. Decision Tree *

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
df = pd.read_csv('/content/drive/MyDrive/brain_stroke.csv')
print(df.head())
```

```
    gender   age  hypertension  heart_disease ever_married      work_type  \
0     Male  67.0             0              1          Yes        Private
1     Male  80.0             0              1          Yes        Private
2   Female  49.0             0              0          Yes        Private
3   Female  79.0             1              0          Yes  Self-employed
4     Male  81.0             0              0          Yes        Private

  Residence_type  avg_glucose_level   bmi   smoking_status  stroke
0          Urban             228.69  36.6  formerly smoked       1
1          Rural             105.92  32.5     never smoked       1
2          Urban             171.23  34.4           smokes       1
3          Rural             174.12  24.0     never smoked       1
4          Urban             186.21  29.0  formerly smoked       1
```

```
d = {'formerly smoked': 0, 'never smoked': 1, 'smokes': 2, 'Unknown' : 3}
df['smoking_status'] = df['smoking_status'].map(d)
print(df.head())
```

```
    gender   age  hypertension  heart_disease ever_married      work_type  \
0     Male  67.0             0              1          Yes        Private
1     Male  80.0             0              1          Yes        Private
2   Female  49.0             0              0          Yes        Private
3   Female  79.0             1              0          Yes  Self-employed
4     Male  81.0             0              0          Yes        Private

  Residence_type  avg_glucose_level   bmi  smoking_status  stroke
0          Urban             228.69  36.6               0       1
1          Rural             105.92  32.5               1       1
2          Urban             171.23  34.4               2       1
3          Rural             174.12  24.0               1       1
4          Urban             186.21  29.0               0       1
```

```
X1 = df[['smoking_status','avg_glucose_level']]
y1 = df['stroke']
print(X1.head())
print(y1.head())
```

```
   smoking_status  avg_glucose_level
0               0             228.69
1               1             105.92
2               2             171.23
3               1             174.12
4               0             186.21
0    1
1    1
2    1
3    1
4    1
Name: stroke, dtype: int64
```

```
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.25, random_state=42)
```

```
dtree3 = DecisionTreeClassifier(max_depth=3)
```

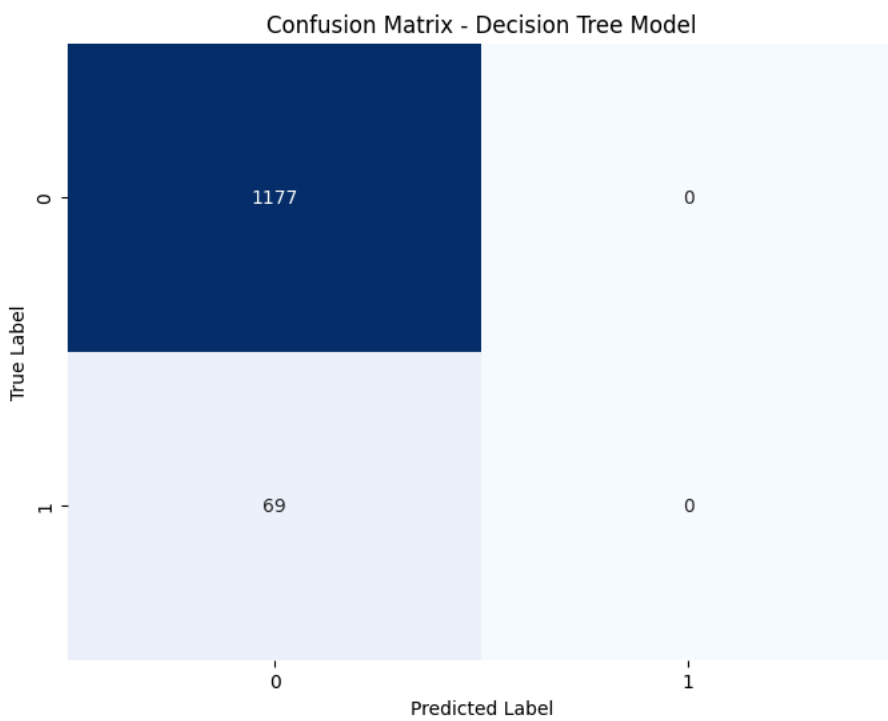
```
dtree3.fit(X1_train, y1_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3)
```

```
y1_pred = dtree3.predict(X1_test)
```

```
cm1 = confusion_matrix(y1_test, y1_pred)
```

```
plt.figure(figsize=(8, 6))
sns.heatmap(cm1, annot=True, fmt='d', cmap='Blues', cbar=False, xticklabels=[0, 1], yticklabels=[0, 1])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix - Decision Tree Model')
plt.show()
```




```
accuracy1 = accuracy_score(y1_test, y1_pred)
print("Accuracy:", accuracy1)
```

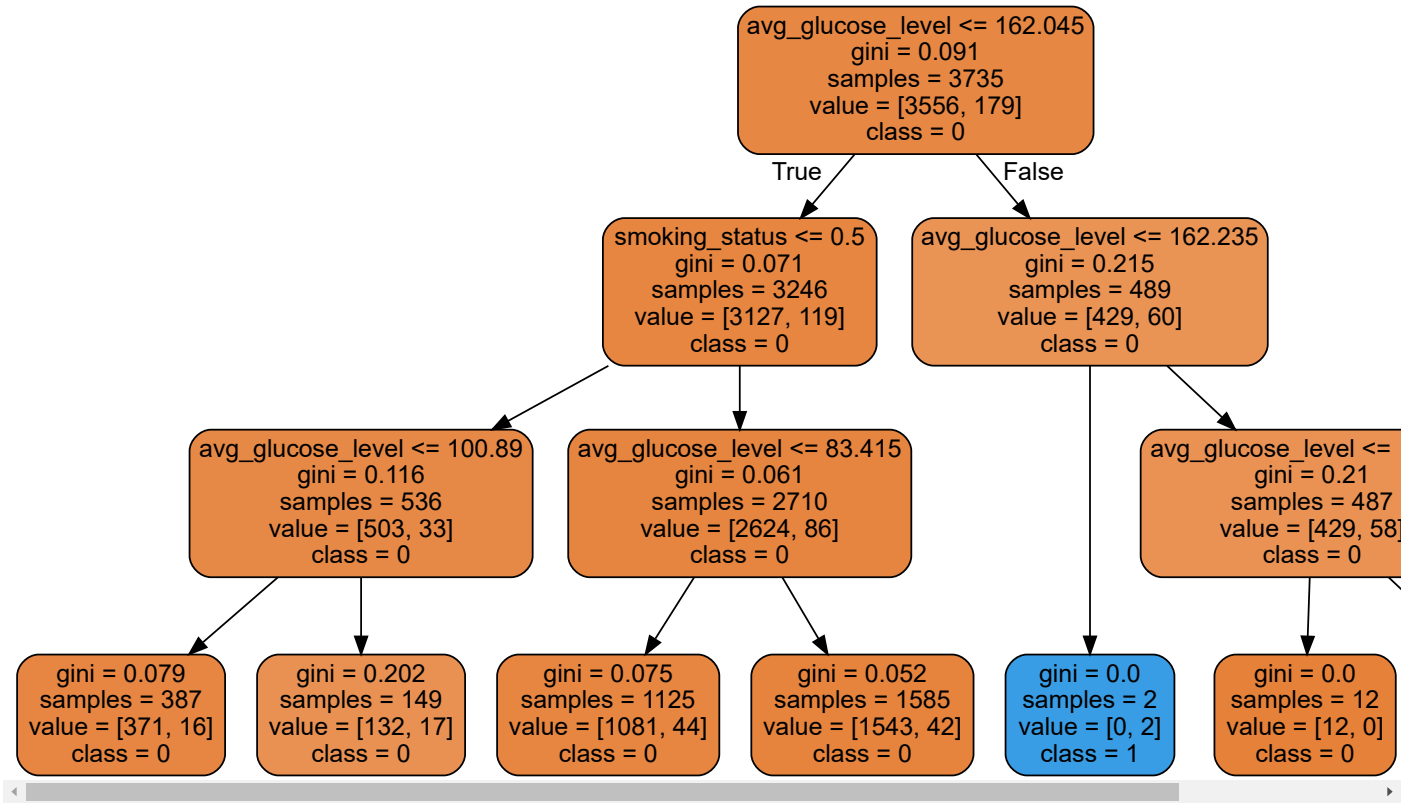
```
Accuracy: 0.9446227929373997
```

```
print(classification_report(y1_test, y1_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      1.00      0.97      1177
           1       0.00      0.00      0.00        69

    accuracy                           0.94      1246
   macro avg       0.47      0.50      0.49      1246
weighted avg       0.89      0.94      0.92      1246

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-d
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-d
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-d
  _warn_prf(average, modifier, msg_start, len(result))
```

```
export_graphviz(decision_tree=dtree3,
out_file='C:\Residence_type.dot',
feature_names=X1_test.columns,
class_names=dtree3.classes_.astype(str),
```

```
leaves_parallel=True,
filled=True,
rotate=False,
rounded=True)
from graphviz import Source
Source.from_file('C:\Residence_type.dot')
```



2. Decision Tree Extention

```
d = {'Yes' : 0, 'No' : 1}
df['ever_married'] = df['ever_married'].map(d)
print(df.head())
```

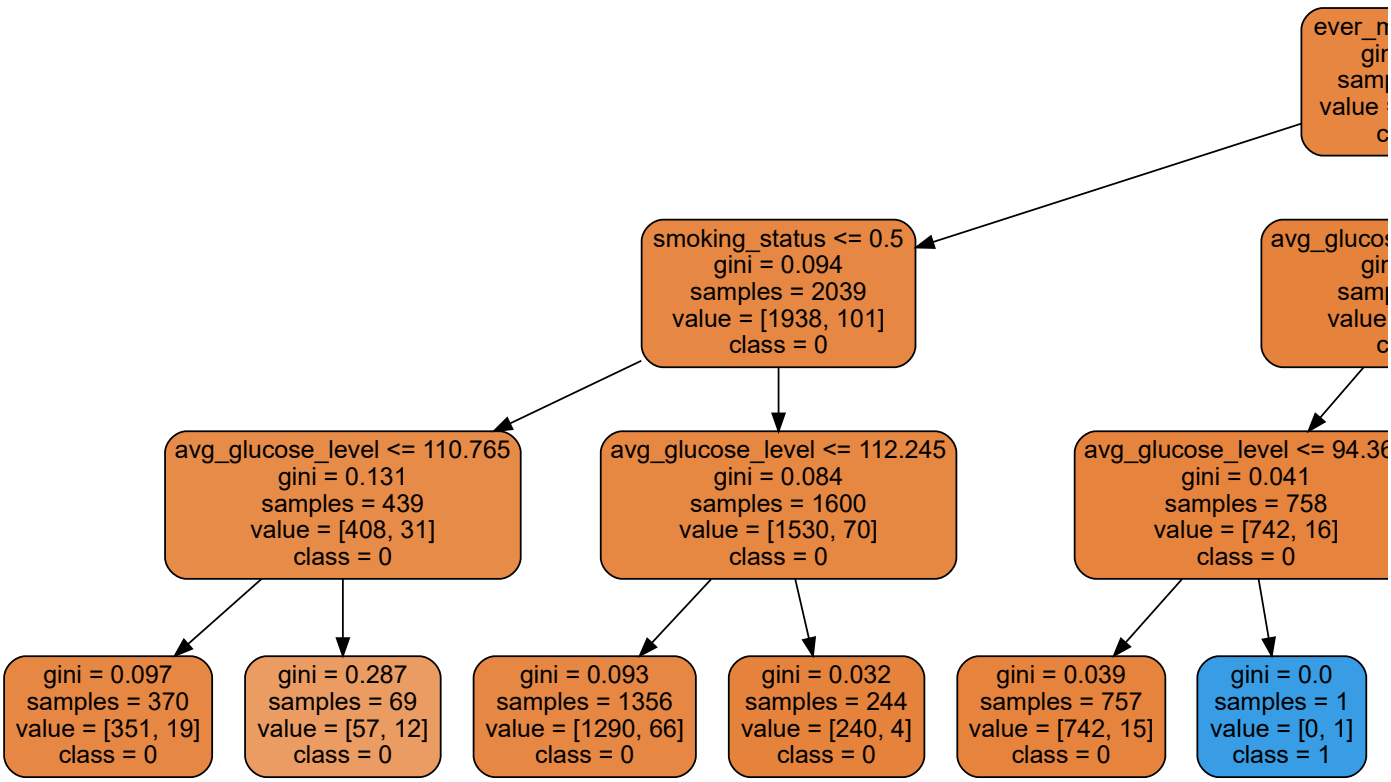
```
    gender   age  hypertension  heart_disease  ever_married      work_type  \
0     Male  67.0             0              1             0        Private
1     Male  80.0             0              1             0        Private
2   Female  49.0             0              0             0        Private
3   Female  79.0             1              0             0  Self-employed
4     Male  81.0             0              0             0        Private

  Residence_type  avg_glucose_level   bmi  smoking_status  stroke
0          Urban             228.69  36.6               0       1
1          Rural             105.92  32.5               1       1
2          Urban             171.23  34.4               2       1
3          Rural             174.12  24.0               1       1
4          Urban             186.21  29.0               0       1
```

```
X2 = df[['smoking_status','avg_glucose_level','ever_married']]
y2 = df['stroke']
print(X2.head())
print(y2.head())
```

```
   smoking_status  avg_glucose_level  ever_married
0               0             228.69             0
1               1             105.92             0
2               2             171.23             0
3               1             174.12             0
4               0             186.21             0
0    1
1    1
2    1
3    1
4    1
Name: stroke, dtype: int64
```

```
X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2, test_size=0.25, random_state=0)
dtree3 = DecisionTreeClassifier(max_depth=4)
dtree3.fit(X2_train, y2_train)
y2_pred = dtree3.predict(X2_test)
```

```
export_graphviz(decision_tree=dtree3,
out_file='C:\Residence_type.dot',
feature_names=X2_test.columns,
class_names=dtree3.classes_.astype(str),
leaves_parallel=True,
filled=True,
rotate=False,
rounded=True)
from graphviz import Source
Source.from_file('C:\Residence_type.dot')
```




```
accuracy2 = accuracy_score(y2_test, y2_pred)
print("Accuracy:", accuracy2)
```

```
Accuracy: 0.9510433386837881
```

LOGISTIC REGRESSION

```
df4 = df.copy()
df4.head()
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	Male	67.0	0	1	0	Private	Urban	228.69	36.6	0	1
1	Male	80.0	0	1	0	Private	Rural	105.92	32.5	1	1
2	Female	49.0	0	0	0	Private	Urban	171.23	34.4	2	1
3	Female	79.0	1	0	0	Self-employed	Rural	174.12	24.0	1	1

Next steps: Generate code with df4 | View recommended plots | New interactive sheet

```
X4 = df4[['avg_glucose_level', 'bmi']]
y4 = df4['ever_married']
X4.head()
```

	avg_glucose_level	bmi
0	228.69	36.6
1	105.92	32.5
2	171.23	34.4
3	174.12	24.0
4	186.21	29.0

Next steps: Generate code with X4 | View recommended plots | New interactive sheet

```
from sklearn.preprocessing import StandardScaler
# Standardizing the data
scaler = StandardScaler()
scaler.fit(X4)
X4_scaled = scaler.transform(X4)
df_scaled = pd.DataFrame(X4_scaled, columns=X4.columns)
df_scaled.head()
```

	avg_glucose_level	bmi
0	2.723411	1.193238
1	-0.000523	0.589390
2	1.448529	0.869222
3	1.512650	-0.662492
4	1.780895	0.073909

Next steps: Generate code with df_scaled | View recommended plots | New interactive sheet

```
X4_train, X4_test, y4_train, y4_test = train_test_split(X4, y4, test_size=0.30, random_state=42)
```

```
from sklearn.linear_model import LogisticRegression
# Building the logistic regression model
logreg = LogisticRegression()
logreg.fit(X4_train, y4_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
# Predicting the target variable
y4_pred = logreg.predict(X4_test)
```

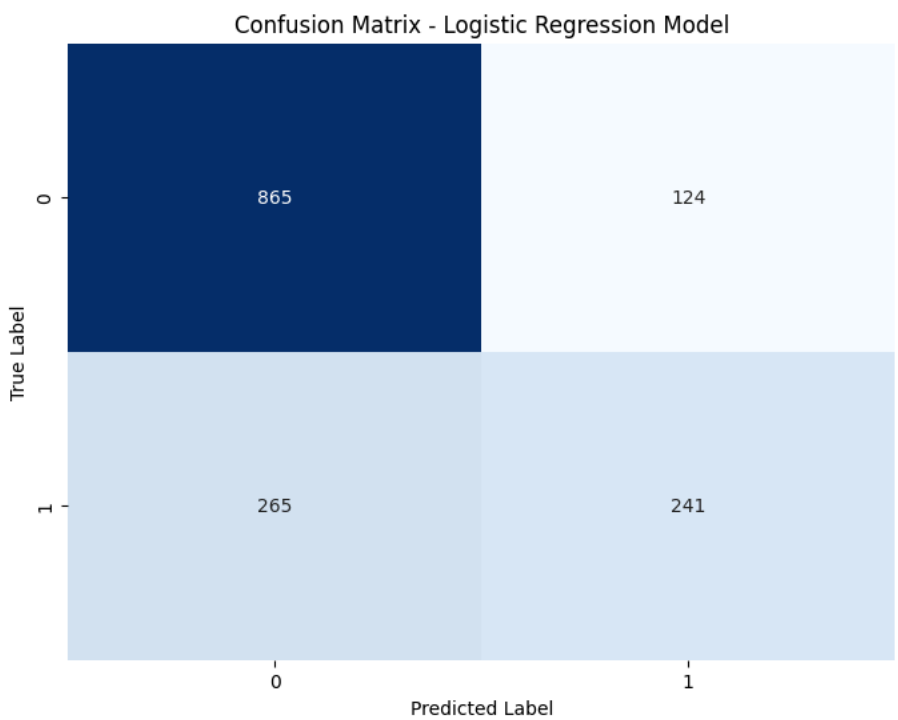
```
# Finding the accuracy score
accuracy4 = accuracy_score(y4_test, y4_pred)
print("Accuracy:", accuracy4)
```

```
Accuracy: 0.7397993311036789
```

```
print(classification_report(y4_test, y4_pred))
```

```
              precision    recall  f1-score   support

           0       0.77      0.87      0.82       989
           1       0.66      0.48      0.55       506

    accuracy                           0.74      1495
   macro avg       0.71      0.68      0.68      1495
weighted avg       0.73      0.74      0.73      1495
```

```
# Calculate the confusion matrix
cm3 = confusion_matrix(y4_test, y4_pred)
# Plot the confusion matrix using seaborn's heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(cm3, annot=True, fmt='d', cmap='Blues', cbar=False, xticklabels=[0, 1], yticklabels=[0, 1])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix - Logistic Regression Model')
plt.show()
```



KNN MODEL

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

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
df = pd.read_csv('/content/drive/MyDrive/brain_stroke.csv')
print(df.head())
```

```
    gender   age  hypertension  heart_disease ever_married      work_type  \
0     Male  67.0             0              1          Yes        Private
1     Male  80.0             0              1          Yes        Private
2   Female  49.0             0              0          Yes        Private
3   Female  79.0             1              0          Yes  Self-employed
4     Male  81.0             0              0          Yes        Private

       Residence_type  avg_glucose_level   bmi   smoking_status  stroke
```

```
0          Urban             228.69  36.6  formerly smoked       1
1          Rural             105.92  32.5     never smoked       1
2          Urban             171.23  34.4           smokes       1
3          Rural             174.12  24.0     never smoked       1
4          Urban             186.21  29.0  formerly smoked       1
```

```
X5 = df[['age', 'bmi']]
y5 = df['hypertension']
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X5)
X5_scaled = scaler.transform(X5)
```

```
df_scaled = pd.DataFrame(X5_scaled, columns=X5.columns)
df_scaled.head()
```

	age	bmi
0	1.040584	1.193238
1	1.614270	0.589390
2	0.246250	0.869222
3	1.570141	-0.662492
4	1.658400	0.073909

Next steps: Generate code with df_scaled | View recommended plots | New interactive sheet

```
from sklearn.model_selection import train_test_split
# Splitting the data into train and test sets:
X5_train, X5_test, y5_train, y5_test = train_test_split(df_scaled, y5, test_size=0.20, random_state=42)
```

```
# Importing the KNeighborsClassifier class from the sklearn.neighbors library:
from sklearn.neighbors import KNeighborsClassifier
```

```
# Building the model
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X5_train, y5_train)
```

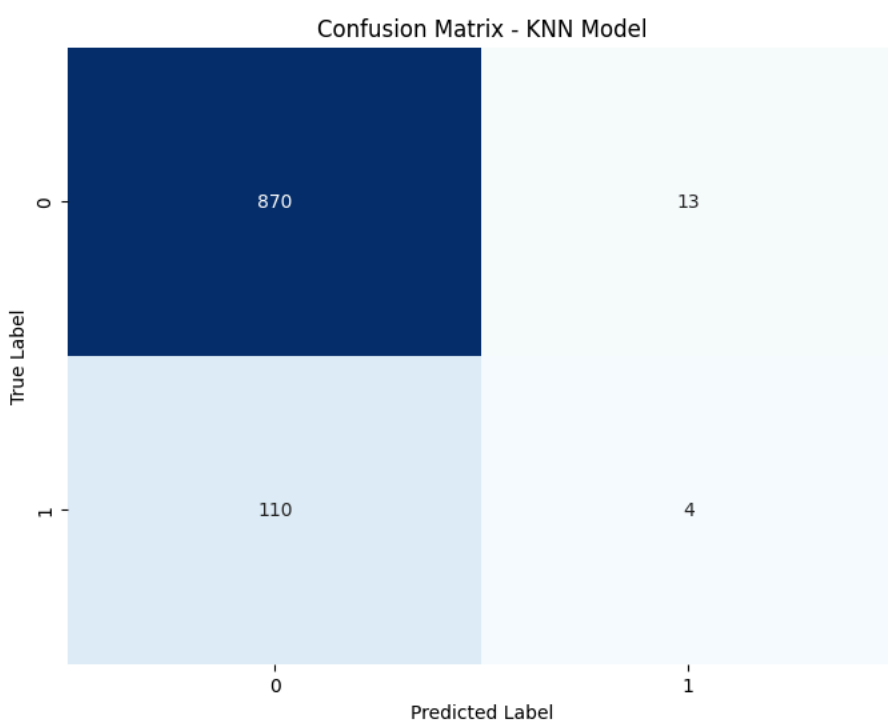
```
▾ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=7)
```

```
# Classifying the records of the X5_test using the model:
y5_pred = knn.predict(X5_test)
```

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

```
cm5 = confusion_matrix(y5_test, y5_pred)
```

```
plt.figure(figsize=(8, 6))
sns.heatmap(cm5, annot=True, fmt='d', cmap='Blues', cbar=False, xticklabels=[0, 1], yticklabels=[0, 1])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix - KNN Model')
plt.show()
```




```
accuracy5 = accuracy_score(y5_test, y5_pred)
print("Accuracy:", accuracy5)
```

```
Accuracy: 0.876629889669007
```

```
print(classification_report(y5_test, y5_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.99      0.93       883
           1       0.24      0.04      0.06       114

    accuracy                           0.88       997
   macro avg       0.56      0.51      0.50       997
weighted avg       0.81      0.88      0.83       997
```