```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.
import kagglehub
iammustafatz_diabetes_prediction_dataset_path = kagglehub.dataset_download('iammustafatz/diabetes-prediction-dataset')

print('Data source import complete.')
```

```
Data source import complete.
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
from sklearn.preprocessing import LabelEncoder , MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

LOADING DATA

```
df = pd.read_csv("/kaggle/input/diabetes-prediction-dataset/diabetes_prediction_dataset.csv")
df
```

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80.0	0	1	never	25.19	6.6	140	0
1	Female	54.0	0	0	No Info	27.32	6.6	80	0
2	Male	28.0	0	0	never	27.32	5.7	158	0
3	Female	36.0	0	0	current	23.45	5.0	155	0
4	Male	76.0	1	1	current	20.14	4.8	155	0
...	...	...	...	...	...	...	...	...	...
99995	Female	80.0	0	0	No Info	27.32	6.2	90	0
99996	Female	2.0	0	0	No Info	17.37	6.5	100	0
99997	Male	66.0	0	0	former	27.83	5.7	155	0
99998	Female	24.0	0	0	never	35.42	4.0	100	0
99999	Female	57.0	0	0	current	22.43	6.6	90	0

100000 rows × 9 columns

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

DATA OVERVIEW

```
df.head()
```

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80.0	0	1	never	25.19	6.6	140	0
1	Female	54.0	0	0	No Info	27.32	6.6	80	0
2	Male	28.0	0	0	never	27.32	5.7	158	0
3	Female	36.0	0	0	current	23.45	5.0	155	0
4	Male	76.0	1	1	current	20.14	4.8	155	0

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

```
df.tail()
```

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
99995	Female	80.0	0	0	No Info	27.32	6.2	90	0
99996	Female	2.0	0	0	No Info	17.37	6.5	100	0
99997	Male	66.0	0	0	former	27.83	5.7	155	0
99998	Female	24.0	0	0	never	35.42	4.0	100	0
99999	Female	57.0	0	0	current	22.43	6.6	90	0

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 9 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   gender               100000 non-null  object 
 1   age                  100000 non-null  float64
 2   hypertension         100000 non-null  int64  
 3   heart_disease        100000 non-null  int64  
 4   smoking_history      100000 non-null  object 
 5   bmi                  100000 non-null  float64
 6   HbA1c_level          100000 non-null  float64
 7   blood_glucose_level  100000 non-null  int64  
 8   diabetes             100000 non-null  int64  
dtypes: float64(3), int64(4), object(2)
memory usage: 6.9+ MB
```

```
df.describe()
```

	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	diabetes
count	100000.000000	100000.00000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000
mean	41.885856	0.07485	0.039420	27.320767	5.527507	138.058060	0.085000
std	22.516840	0.26315	0.194593	6.636783	1.070672	40.708136	0.278883
min	0.080000	0.00000	0.000000	10.010000	3.500000	80.000000	0.000000
25%	24.000000	0.00000	0.000000	23.630000	4.800000	100.000000	0.000000
50%	43.000000	0.00000	0.000000	27.320000	5.800000	140.000000	0.000000
75%	60.000000	0.00000	0.000000	29.580000	6.200000	159.000000	0.000000
max	80.000000	1.00000	1.000000	95.690000	9.000000	300.000000	1.000000

```
df.describe(include="O")
```

	gender	smoking_history
count	100000	100000
unique	3	6
top	Female	No Info
freq	58552	35816

```
df.shape
```

```
(100000, 9)
```

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

	0
gender	0
age	0
hypertension	0
heart_disease	0
smoking_history	0
bmi	0
HbA1c_level	0
blood_glucose_level	0
diabetes	0

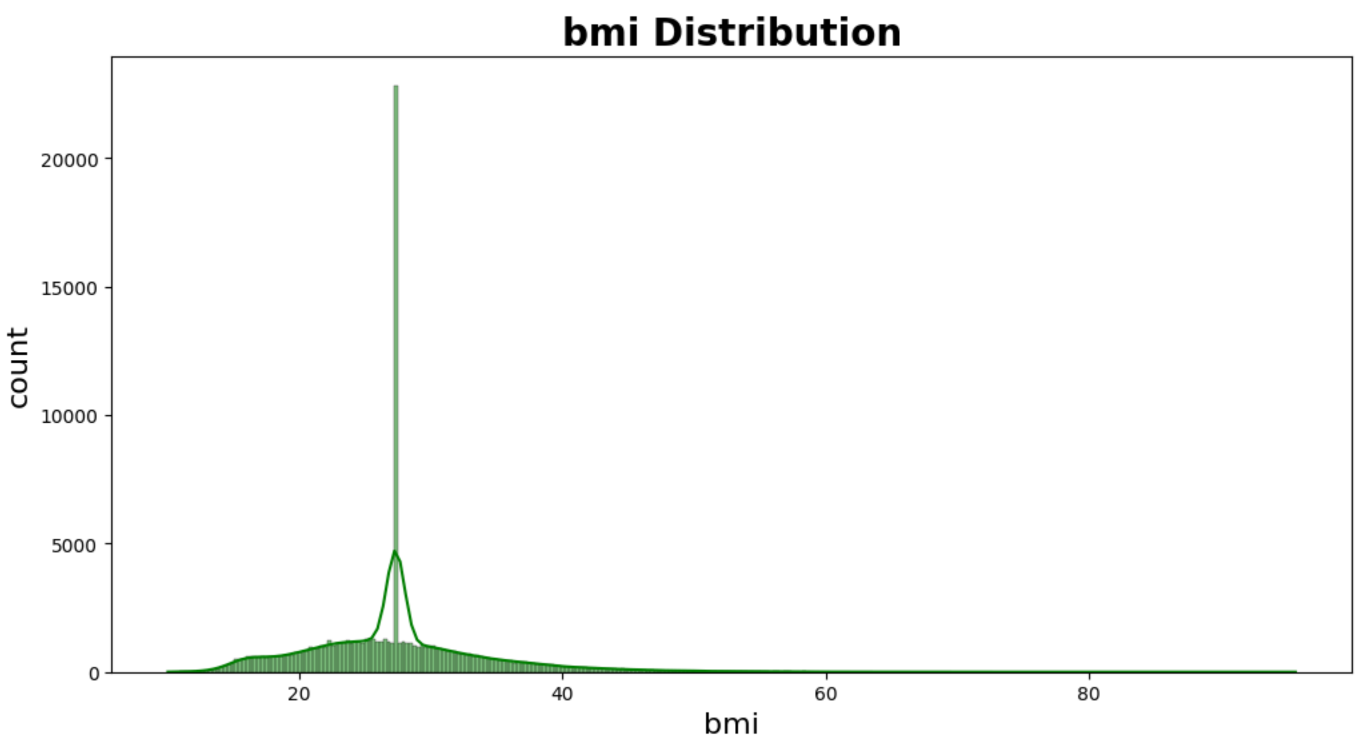
dtype: int64

```
df.duplicated().sum()
```

```
np.int64(3854)
```

```
df.drop_duplicates(inplace=True)
```

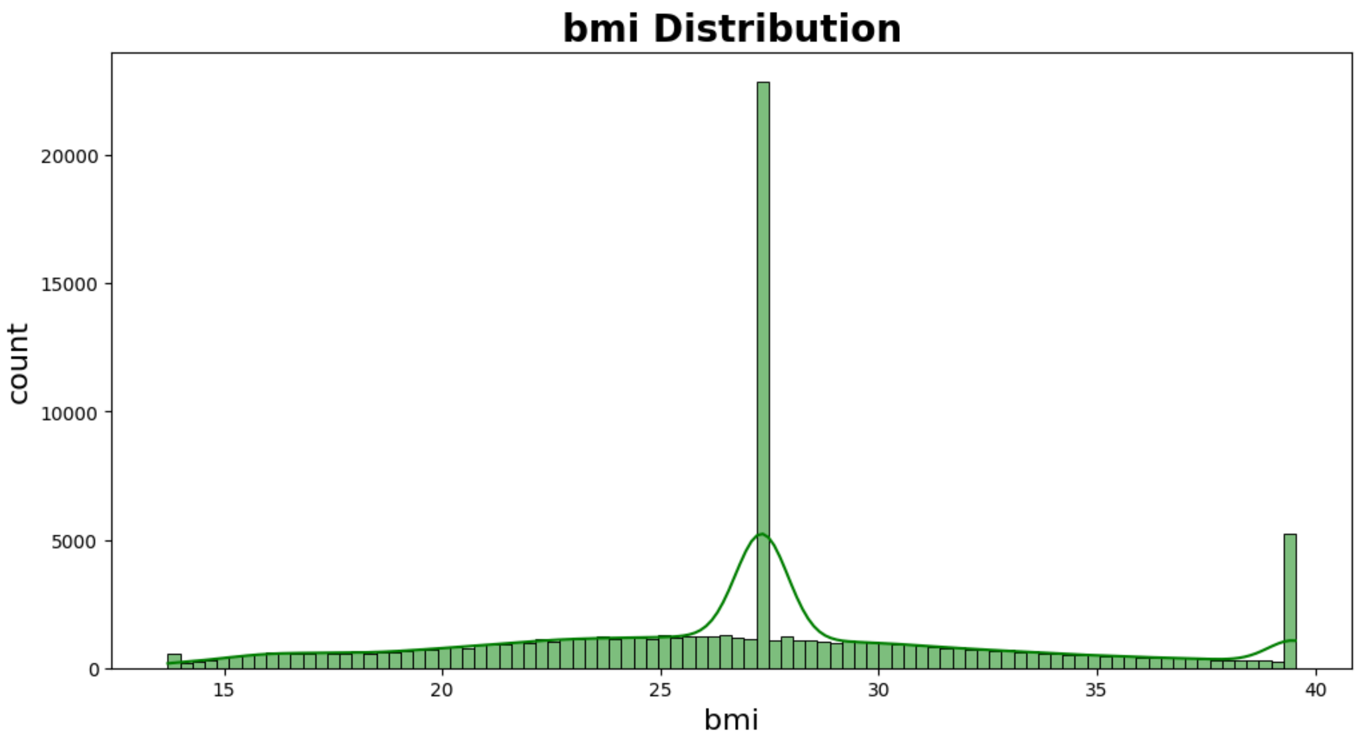
```
plt.figure(figsize=(12,6))
sns.histplot(x=df['bmi'],kde=True,color="green")
plt.title('bmi Distribution',fontsize=20,fontweight='bold')
plt.xlabel('bmi',fontsize=16)
plt.ylabel('count',fontsize=16)
plt.show()
```



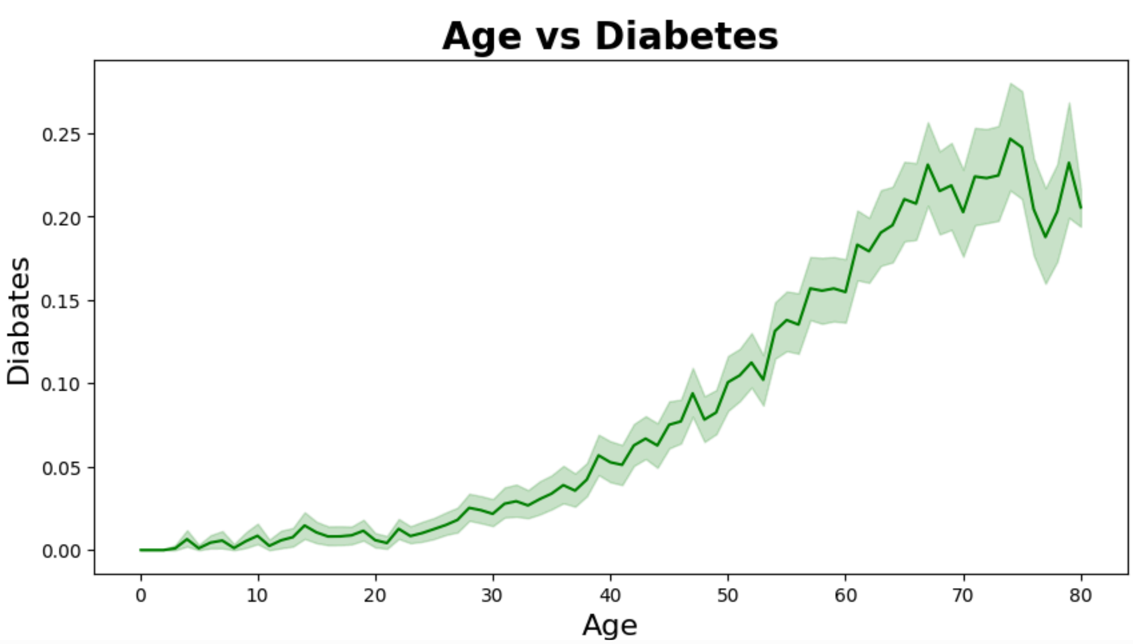

```
q1 = df['bmi'].quantile(0.25)
q3 = df['bmi'].quantile(0.75)
iqr = q3 - q1
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr
df['bmi']=df['bmi'].clip(lower_bound,upper_bound)
```

```
plt.figure(figsize=(12,6))
sns.histplot(x=df['bmi'],kde=True,color="green")
plt.title('bmi Distribution',fontsize=20,fontweight='bold')
```

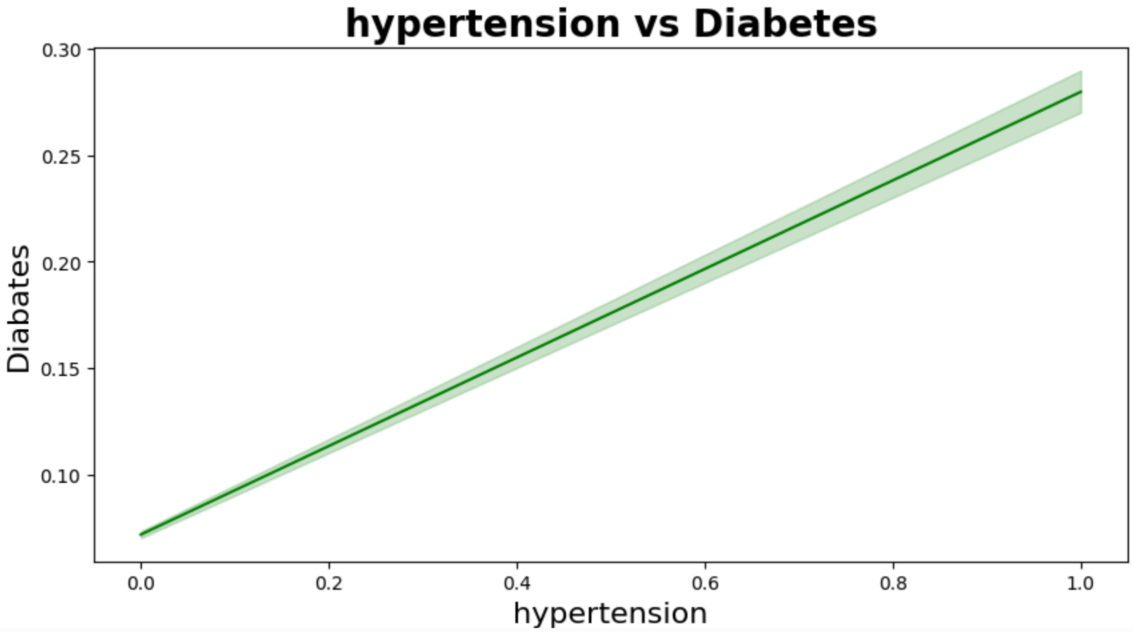
```
plt.xlabel('bmi',fontsize=16)
plt.ylabel('count',fontsize=16)
plt.show()
```




```
plt.figure(figsize=(10,5))
sns.lineplot(x=df['age'],y=df['diabetes'],color='green')
plt.title('Age vs Diabetes',fontsize=20,fontweight='bold')
plt.xlabel('Age',fontsize=16)
plt.ylabel('Diabates',fontsize=16)
plt.show()
```




```
plt.figure(figsize=(10,5))
sns.lineplot(x=df['hypertension'],y=df['diabetes'],color='green')
plt.title('hypertension vs Diabetes',fontsize=20,fontweight='bold')
plt.xlabel('hypertension',fontsize=16)
plt.ylabel('Diabates',fontsize=16)
plt.show()
```



CONVERT COLUMNS INTO NUMERIC

```
le = LabelEncoder()
df['gender']= le.fit_transform(df['gender'])
```

```
df["gender"]
```

	gender
0	0
1	0
2	1
3	0
4	1
...	...
99994	0
99996	0
99997	1
99998	0
99999	0

96146 rows × 1 columns

dtype: int64

```
le = LabelEncoder()
df['smoking_history']= le.fit_transform(df['smoking_history'])
```

```
df["smoking_history"]
```

	smoking_history
0	4
1	0
2	4
3	1
4	1
...	...
99994	0
99996	0
99997	3
99998	4
99999	1

96146 rows × 1 columns

dtype: int64

```
plt.figure(figsize=(25, 10))
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True,cmap='Purples', fmt=".2f")
plt.title("correlation Matix")
plt.show()
```

SCALLING

```
numerical_cols = ['age','bmi', 'HbA1c_level', 'blood_glucose_level']
```

```
scaler = MinMaxScaler()

df[numerical_cols] = scaler.fit_transform(df[numerical_cols])
```

SPILT DATA

```
x = df.drop('diabetes',axis=1)
y = df['diabetes']
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y, test_size = 0.2, random_state=42)
```

LOGISTIC REGRESSION MODEL

```
model = LogisticRegression()
model.fit(x_train,y_train)
```

```
▸ LogisticRegression
```

ACCURACY

```
y_pred = model.predict(x_test)

print("Accuracy : ",accuracy_score(y_test,y_pred))
```

```
Accuracy :  0.9573062922516901
```

```
print(model.score(x_train,y_train))
print(model.score(x_test,y_test))
```

```
0.9593062561755682
0.9573062922516901
```

```
print(classification_report(y_test,y_pred))
```

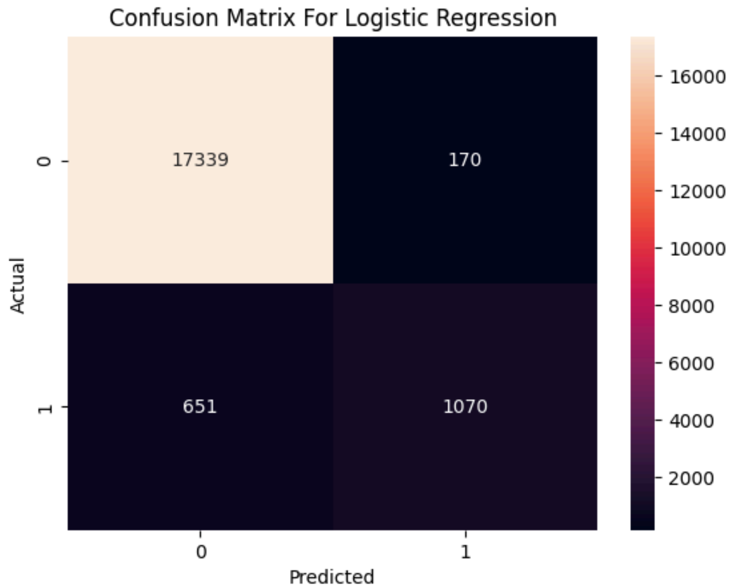
```
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     17509
           1       0.86      0.62      0.72      1721

    accuracy                           0.96     19230
   macro avg       0.91      0.81      0.85     19230
weighted avg       0.95      0.96      0.95     19230
```

CONFUSION MATRIX

```
sns.heatmap(confusion_matrix(y_test,y_pred) , annot=True, fmt='d' )
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix For Logistic Regression')
plt.show()
```



SAVING MODEL

```
import pickle

pickle.dump(model,open("Logistic_Model.pkl","wb"))
```

SVC MODEL

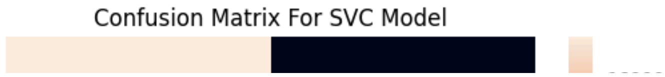
```
svc_model=SVC()
svc_model.fit(x_train,y_train)
```

```
▾ SVC
SVC()
```

```
y_pred_svc=svc_model.predict(x_test)
print(accuracy_score(y_pred_svc, y_test))
```

```
0.9583463338533541
```

```
sns.heatmap(confusion_matrix(y_test, y_pred_svc), annot=True, fmt='d')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix For SVC Model')
plt.show()
```

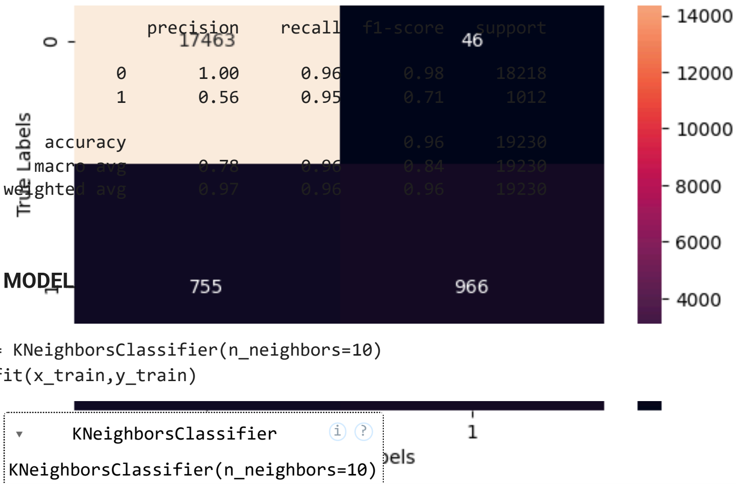



```
print(classification_report(y_pred_svc, y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98     18218
           1       0.56      0.95      0.71      1012

    accuracy                           0.96     19230
   macro avg       0.78      0.96      0.84     19230
weighted avg       0.97      0.96      0.96     19230
```



KNN MODEL

```
KNN = KNeighborsClassifier(n_neighbors=10)
KNN.fit(x_train,y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=10)
```

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

```
print("Accuracy : ",accuracy_score(y_test,y_pred))
```

```
Accuracy :  0.9571502860114405
```

```
print(KNN.score(x_train,y_train))
print(KNN.score(x_test,y_test))
```

```
0.9617374798481461
0.9571502860114405
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     17509
           1       0.96      0.54      0.69      1721

    accuracy                           0.96     19230
   macro avg       0.96      0.77      0.84     19230
weighted avg       0.96      0.96      0.95     19230
```

```
sns.heatmap( confusion_matrix(y_test,y_pred) , annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix For KNN Model')
plt.show()
```