Importing Libraries

```
#@title Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
```

```
# Read your dataset
data=pd.read_csv("/content/drive/MyDrive/diabetes_prediction_dataset.csv")
```

```
data.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 data | View recommended plots | New interactive sheet

```
data.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

```
data.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
```

```
data['gender'].unique()
```

```
array(['Female', 'Male', 'Other'], dtype=object)
```

```
from sklearn.preprocessing import OneHotEncoder
columns_to_encode = ['gender', 'smoking_history']

# Create OneHotEncoder instance
encoder = OneHotEncoder(sparse_output=False)

# Fit and transform the data
encoded_data = encoder.fit_transform(data[columns_to_encode])

# Convert the encoded data to a DataFrame with appropriate column names
encoded_df = pd.DataFrame(encoded_data, columns=encoder.get_feature_names_out(columns_to_encode))
```

```
# Concatenate the original dataframe (excluding the original categorical columns) with the encoded columns
df = pd.concat([data.drop(columns_to_encode, axis=1), encoded_df], axis=1)
```

```
df.describe()
```

bmi	HbA1c_level	blood_glucose_level	diabetes	gender_Female
100000.000000	100000.000000	100000.000000	100000.000000	100000.000000
27.320767	5.527507	138.058060	0.085000	0.585520
6.636783	1.070672	40.708136	0.278883	0.492635
10.010000	3.500000	80.000000	0.000000	0.000000
23.630000	4.800000	100.000000	0.000000	0.000000
27.320000	5.800000	140.000000	0.000000	1.000000
29.580000	6.200000	159.000000	0.000000	1.000000
95.690000	9.000000	300.000000	1.000000	1.000000

```
df.corr()
```

	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	diabetes	gender_Female
age	1.000000	0.251171	0.233354	0.337396	0.101354	0.110672	0.258008	0.030480
hypertension	0.251171	1.000000	0.121262	0.147666	0.080939	0.084429	0.197823	-0.014318
heart_disease	0.233354	0.121262	1.000000	0.061198	0.067589	0.070066	0.171727	-0.077832
bmi	0.337396	0.147666	0.061198	1.000000	0.082997	0.091261	0.214357	0.023016
HbA1c_level	0.101354	0.080939	0.067589	0.082997	1.000000	0.166733	0.400660	-0.020015
blood_glucose_level	0.110672	0.084429	0.070066	0.091261	0.166733	1.000000	0.419558	-0.017200
diabetes	0.258008	0.197823	0.171727	0.214357	0.400660	0.419558	1.000000	-0.037553
gender_Female	0.030480	-0.014318	-0.077832	0.023016	-0.020015	-0.017200	-0.037553	1.000000
gender_Male	-0.030282	0.014423	0.077911	-0.023021	0.020058	0.017189	0.037666	-0.999629
gender_Other	-0.007348	-0.003816	-0.002718	0.000119	-0.001528	0.000457	-0.004090	-0.015948
smoking_history_No Info	-0.276945	-0.117210	-0.052398	-0.222553	-0.045979	-0.051635	-0.118939	-0.053810
smoking_history_current	0.030946	0.017930	0.007604	0.053617	0.006123	0.012447	0.019606	-0.026517
smoking_history_ever	0.065768	0.023124	0.040671	0.044343	0.009655	0.006267	0.024080	-0.011019
smoking_history_former	0.216481	0.083401	0.095194	0.111312	0.037306	0.040688	0.097917	-0.048927
smoking_history_never	0.065498	0.045953	-0.030843	0.086849	0.010684	0.012475	0.027267	0.098682
smoking_history_not current	0.067663	0.000997	0.007922	0.034979	0.009830	0.008613	0.020734	0.011419

```
df.isna().sum()
```

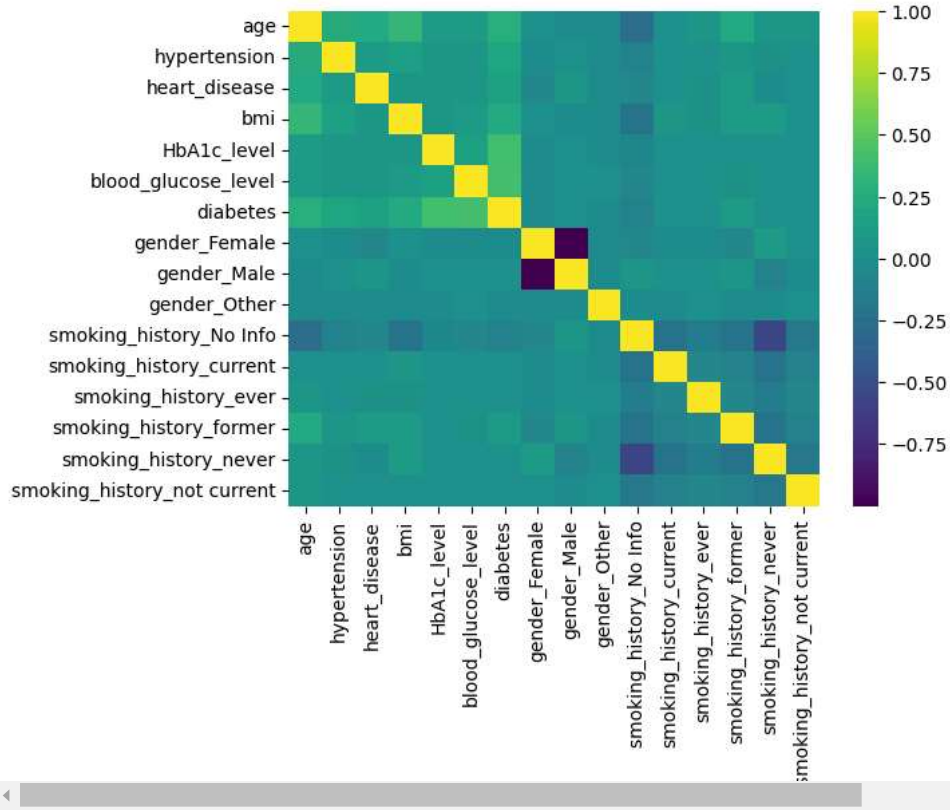
	0
age	0
hypertension	0
heart_disease	0
bmi	0
HbA1c_level	0
blood_glucose_level	0
diabetes	0
gender_Female	0
gender_Male	0
gender_Other	0
smoking_history_No Info	0
smoking_history_current	0
smoking_history_ever	0
smoking_history_former	0
smoking_history_never	0
smoking_history_not current	0

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

	0
age	0
hypertension	0
heart_disease	0
bmi	0
HbA1c_level	0
blood_glucose_level	0
diabetes	0
gender_Female	0
gender_Male	0
gender_Other	0
smoking_history_No Info	0
smoking_history_current	0
smoking_history_ever	0
smoking_history_former	0
smoking_history_never	0
smoking_history_not current	0

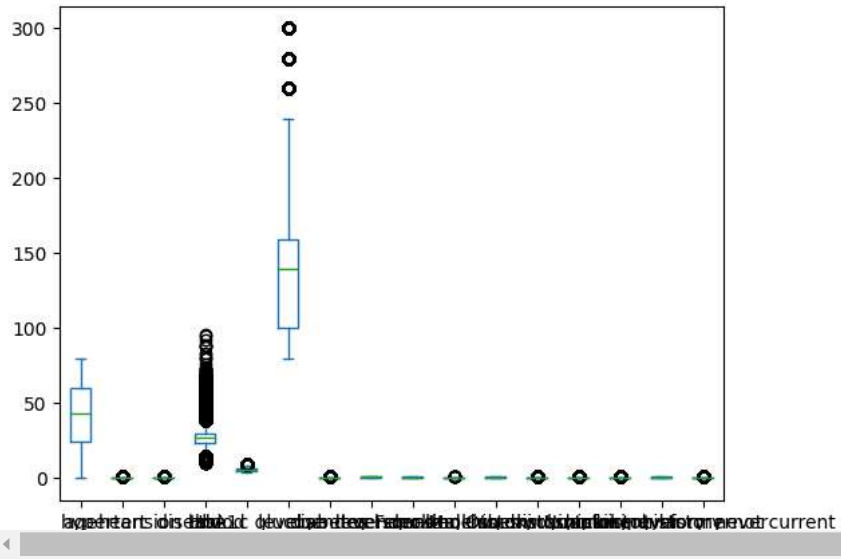
```
figure(figsize=())

sns.heatmap(df.corr(),cmap='viridis')
```

```
<Axes: >
```




```
df.plot(kind='box')
```

```
<Axes: >
```




```
y=df["diabetes"]
df.drop("diabetes",axis=1,inplace=True)
x=df
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
```

```
from sklearn.linear_model import LogisticRegression
```

Logistic Regression

```
#@title Logistic Regression
model=LogisticRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed to converge (status=
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
from sklearn.metrics import accuracy_score, confusion_matrix,f1_score,roc_curve, auc,precision_score, recall_score
```

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

```
Accuracy:  0.96105
```

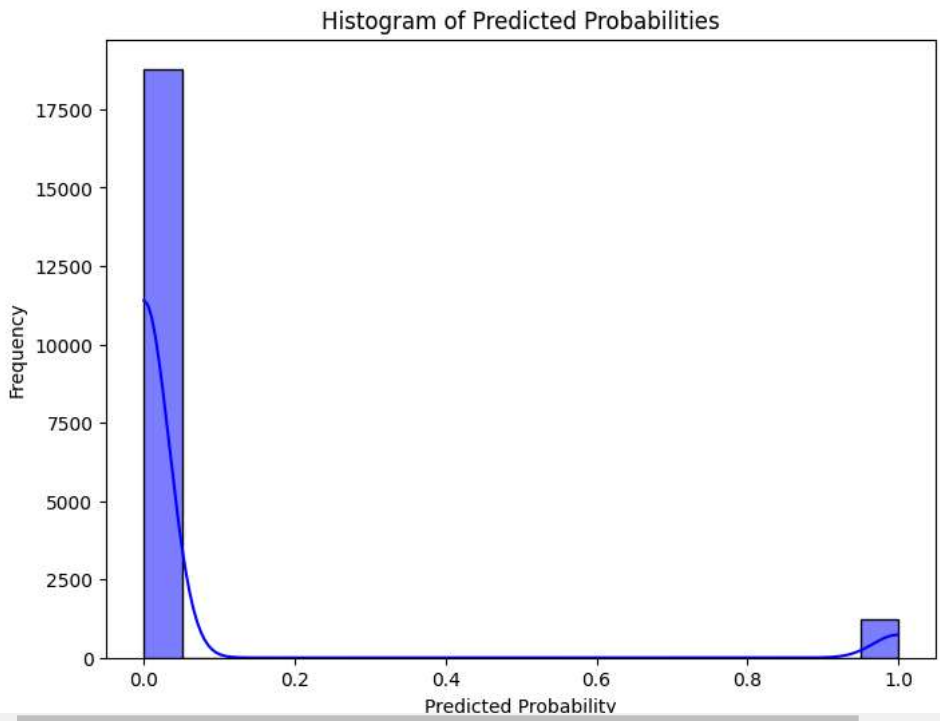
```
confusion_matrix(y_test,y_pred)
```

```
array([[18148,   136],
       [  643,  1073]])
```

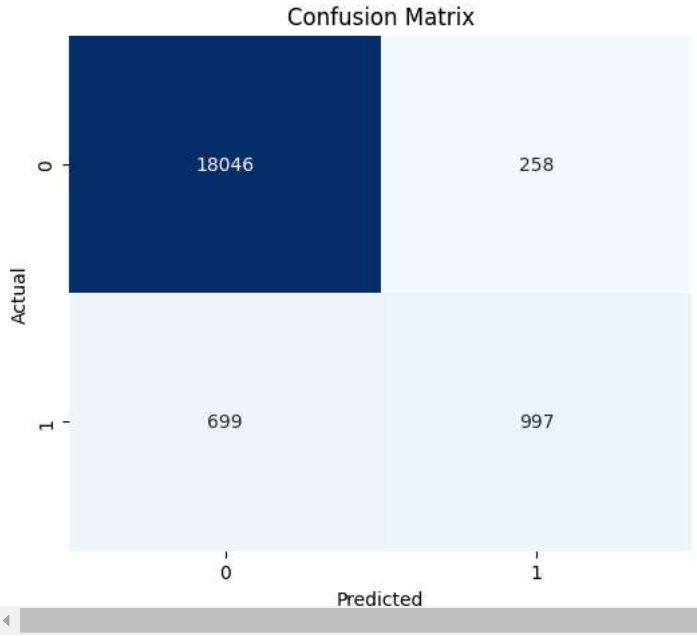
```
print("F1_score: ",f1_score(y_test,y_pred))
```

```
F1_score:  0.7336752136752137
```

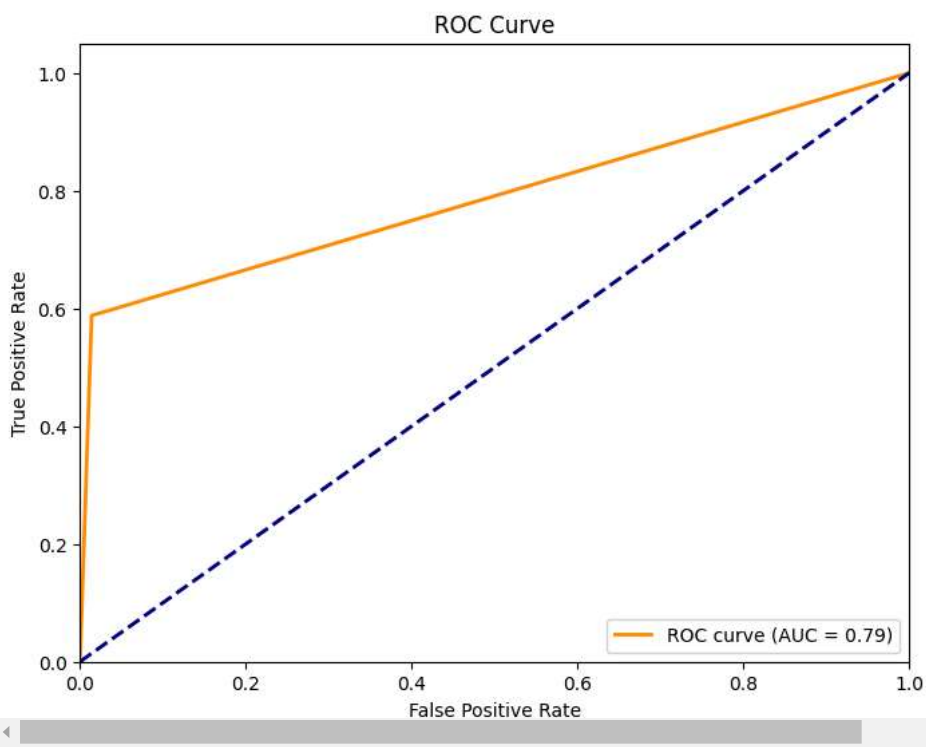
```
plt.figure(figsize=(8, 6))
sns.histplot(y_pred, kde=True, color='blue', bins=20)
plt.title('Histogram of Predicted Probabilities')
plt.xlabel('Predicted Probability')
plt.ylabel('Frequency')
plt.show()
```




```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

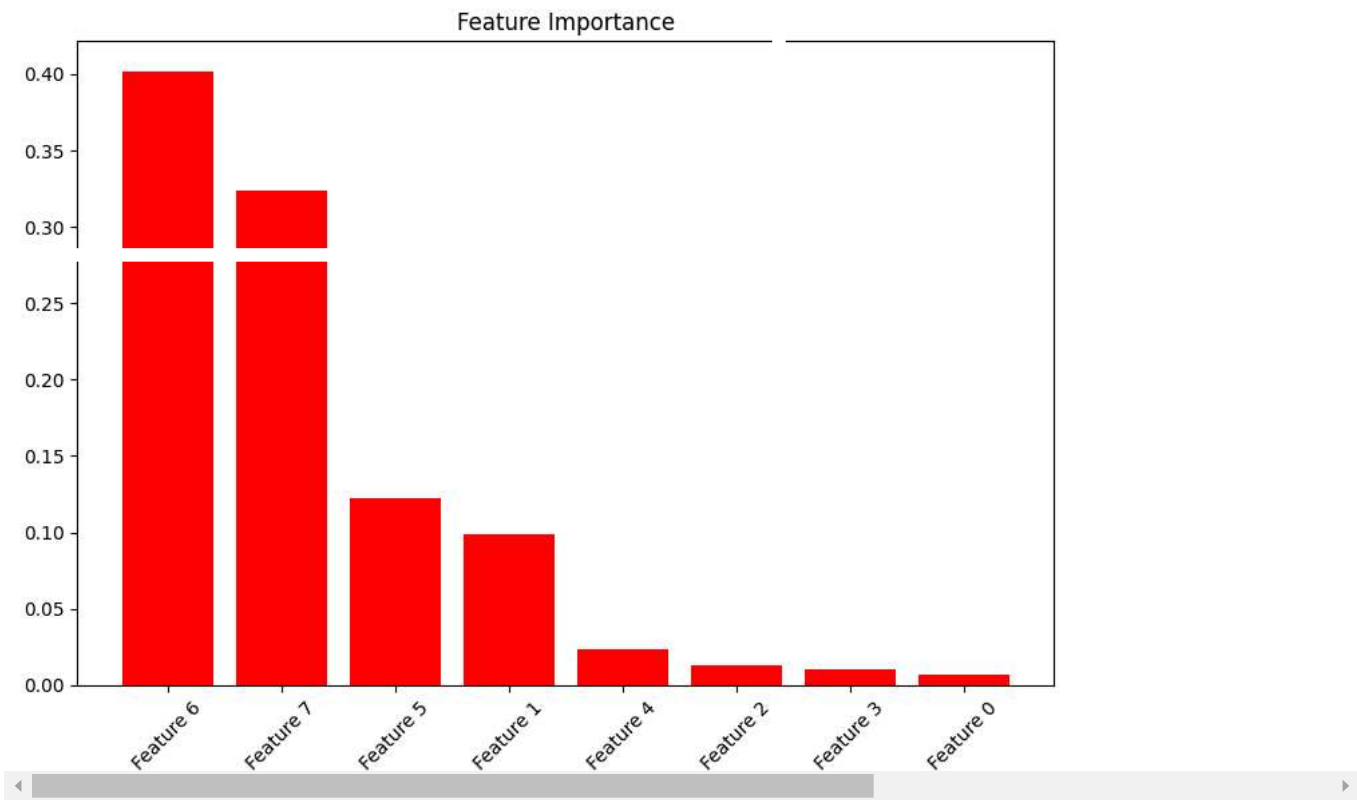



```
fpr, tpr, _ = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.title('ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.show()
```




```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier()
rf_model.fit(x_train, y_train)
importances = rf_model.feature_importances_
indices = np.argsort(importances)[::-1]
features = [f'Feature {i}' for i in range(x.shape[1])]

plt.figure(figsize=(8, 6))
plt.title('Feature Importance')
plt.bar(range(x.shape[1]), importances[indices], color="r", align="center"
```

```
plt.xticks(range(x.shape[1]), [features[i] for i in indices], rotation=45)
plt.tight_layout()
plt.show()
```



Naive Bayes

```
#@title Naive Bayes
from sklearn.naive_bayes import GaussianNB
model2 = GaussianNB()
model2.fit(x_train, y_train)

# Step 4: Make predictions on the test set
y_pred2 = model2.predict(x_test)


from sklearn.metrics import accuracy_score,precision_score, recall_score, f1_score,confusion_matrix
accuracy = accuracy_score(y_test, y_pred2)
precision = precision_score(y_test, y_pred2)
recall = recall_score(y_test, y_pred2)
f1 = f1_score(y_test, y_pred2)
conf_matrix = confusion_matrix(y_test, y_pred2)

# Display the evaluation metrics
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
print("Confusion Matrix:")
print(conf_matrix)
```

```
Accuracy: 0.86
Precision: 0.35
Recall: 0.79
F1 Score: 0.49
Confusion Matrix:
[[15791  2493]
 [  362  1354]]
```

Cross Validation Method

```
#@title Cross Validation Method
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
```