https://drive.google.com/file/d/1v6ouvJyuzhZ80-HkZ96EoUjkt6xEUA2G/view?usp=drive_link

Pima Indians Diabetes Dataset.csv

```
import pandas as pd
import numpy as np
from google.colab import drive
drive.mount('/content/drive')
```

```
df = pd.read_csv("/content/drive/MyDrive/AI Lab(SE334)/Pima Indians Diabetes Dataset.csv")
```

Double-click (or enter) to edit

```
| Column
| Type  | Status
|
| ---------------------- | ----- | ------------------- |
| Pregnancies
| int   | correct
| Glucose
| BloodPressure
| SkinThickness
| Insulin
| BMI
| int   | correct
| int   | correct
| int   | correct
| int   | correct
| DiabetesPedigreeFunction | float | correct
| Age
| Outcome
|
|
|
|
|
| float | correct
| int   | correct
|
|
|
| int   | correct (binary 0/1) |
```

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

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

```
#Drop rows with any missing values
df = df.dropna()
```

```
df = df.drop_duplicates()
df = df.convert_dtypes()
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  ---  ------                    -------
 0   Pregnancies               768 non-null    Int64
 1   Glucose                   768 non-null    Int64
 2   BloodPressure             768 non-null    Int64
 3   SkinThickness             768 non-null    Int64
 4   Insulin                   768 non-null    Int64
 5   BMI                       768 non-null    Float64
 6   DiabetesPedigreeFunction  768 non-null    Float64
 7   Age                       768 non-null    Int64
 8   Outcome                   768 non-null    Int64
dtypes: Float64(2), Int64(7)
memory usage: 60.9 KB
None
```

```
print(df.columns)
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

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

```
0
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
0
0
0
0
0
0
DiabetesPedigreeFunction 0
Age
0
Outcome
dtype: int64
0
```

```
Start coding or generate with AI.
```

No encoding as all data are numarical

```
#featurescalingwithstandarscale
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

```
X = df.drop('Outcome', axis=1)
y = df['Outcome']
```

Feature scaling

```
scaled_x = scaler.fit_transform(X)
scaled_x
```

```
array([[ 0.63994726,  0.84832379,  0.14964075, ...,  0.20401277,
0.46849198,  1.4259954 ],
[-0.84488505, -1.12339636, -0.16054575, ..., -0.68442195,-0.36506078, -0.19067191],
[ 1.23388019,  1.94372388, -0.26394125, ..., -1.10325546,
0.60439732, -0.10558415],
...,
[ 0.3429808 ,  0.00330087,  0.14964075, ..., -0.73518964,-0.68519336, -0.27575966],
[-0.84488505,  0.1597866 , -0.47073225, ..., -0.24020459,-0.37110101,  1.17073215],
[-0.84488505, -0.8730192 ,  0.04624525, ..., -0.20212881,-0.47378505, -0.87137393]])
```

```
Start coding or generate with AI.
```

Train_test split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(scaled_x, y, test_size=0.2, random_state
```

Logestic Regression

```
from sklearn.linear_model import LogisticRegression
LR=LogisticRegression()
```

```
LR.fit(X_train,y_train)
```

```
 LogisticRegression
LogisticRegression()
```

Model Analysis of LR

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
y_pred = LR.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm,annot=True)
```

```
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

```
#presion,f1,recall
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.81      0.80      0.81        99
         1.0       0.65      0.67      0.66        55

    accuracy                           0.75       154
   macro avg       0.73      0.74      0.73       154
weighted avg       0.76      0.75      0.75       154
```

```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange')
```

```
from sklearn.model_selection import learning_curve
train_sizes, train_scores, test_scores = learning_curve(LR, scaled_x, y, cv=5, scoring='accu
plt.plot(train_sizes, np.mean(train_scores, axis=1), label='Training score')
plt.plot(train_sizes, np.mean(test_scores, axis=1), label='Cross-validation score')
plt.title('Learning Curve')
plt.xlabel('Training examples')
plt.ylabel('Score')
plt.legend()
plt.show()
```

```
#Histograms (feature distribution)
df.hist(figsize=(10, 10))
plt.show()
```

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, color='blue')
plt.xlabel('Actual Target Values')
plt.ylabel('Predicted Target Values')
plt.title('Actual vs Predicted (Logestic Regression)')
plt.show()
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
DT=DecisionTreeClassifier()
DT.fit(X_train,y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
Start coding or generate with AI.
```

Model Analysis of DT

```
#confusion_matrix
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
y_pred = DT.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm,annot=True)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

```
#presion,f1,recall
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.83      0.78      0.80        99
         1.0       0.64      0.71      0.67        55

    accuracy                           0.75       154
   macro avg       0.73      0.74      0.74       154
weighted avg       0.76      0.75      0.76       154
```

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
AILAB_1111.ipynb - Colab
y_prob = DT.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', label=f"AUC = {roc_auc:.2f}")
plt.plot([0,1], [0,1], linestyle='--')  # baseline
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Decision Tree ROC Curve")
plt.legend()
plt.show()
```

```
from sklearn.model_selection import learning_curve
train_sizes, train_scores, test_scores = learning_curve(DT, scaled_x, y, cv=5, scoring='accu
plt.plot(train_sizes, np.mean(train_scores, axis=1), label='Training score')
plt.plot(train_sizes, np.mean(test_scores, axis=1), label='Cross-validation score')
plt.title('Learning Curve')
plt.xlabel('Training examples')
plt.ylabel('Score')
plt.legend()
plt.show()
```

```
Start coding or generate with AI.
```

KNN

```
from sklearn.neighbors import KNeighborsClassifier
KNN = KNeighborsClassifier(n_neighbors=5)
KNN.fit(X_train, y_train)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
i
```

```
#confusion_matrix
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
y_pred = KNN.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm,annot=True)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.74      0.80      0.77        99
         1.0       0.57      0.49      0.53        55

    accuracy                           0.69       154
   macro avg       0.66      0.64      0.65       154
weighted avg       0.68      0.69      0.68       154
```

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
# Get probability for class 1 (required for ROC)
y_prob = KNN.predict_proba(X_test)[:, 1]
# ROC calculation
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)
# Plot ROC Curve
plt.figure()
plt.plot(fpr, tpr, color='darkorange', label=f"AUC = {roc_auc:.2f}")
plt.plot([0,1], [0,1], linestyle='--')  # baseline
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("KNN ROC Curve")
plt.legend()
plt.show()
```

```
from sklearn.model_selection import learning_curve
import matplotlib.pyplot as plt
train_sizes, train_scores, test_scores = learning_curve(
KNN,
# <-- KNN model
scaled_x,
y,
# <-- your features
# <-- your labels
cv=5,
scoring='accuracy',
train_sizes=np.linspace(0.1, 1.0, 10)
)
plt.plot(train_sizes, np.mean(train_scores, axis=1), label='Training score')
plt.plot(train_sizes, np.mean(test_scores, axis=1), label='Cross-validation score')
plt.title('KNN Learning Curve')
```

```
plt.xlabel('Training examples')
plt.ylabel('Score')
plt.legend()
plt.show()
```

```
from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier(n_estimators=100, random_state=42)
RF.fit(X_train, y_train)
▾
RandomForestClassifier
i
?
RandomForestClassifier(random_state=42)
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
# Predict using Random Forest
y_pred = RF.predict(X_test)
# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)
# Plot
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Random Forest Confusion Matrix")
plt.show()
```

```
from sklearn.metrics import classification_report
# Make sure y_pred is predicted from Random Forest
y_pred = RF.predict(X_test)
# Print precision, recall, f1-score
print(classification_report(y_test, y_pred))
              precision    recall  f1-score   support

         0.0       0.79      0.79      0.79        99
         1.0       0.62      0.62      0.62        55

    accuracy                           0.73       154
   macro avg       0.70      0.70      0.70       154
weighted avg       0.73      0.73      0.73       154
```

```
from sklearn.model_selection import learning_curve
import matplotlib.pyplot as plt
import numpy as np
train_sizes, train_scores, test_scores = learning_curve(
RF,
# <-- Random Forest model
scaled_x,
y,
# <-- your features
# <-- your labels
cv=5,
scoring='accuracy',
train_sizes=np.linspace(0.1, 1.0, 10)
)
plt.plot(train_sizes, np.mean(train_scores, axis=1), label='Training score')
plt.plot(train_sizes, np.mean(test_scores, axis=1), label='Cross-validation score')
plt.title('Random Forest Learning Curve')
plt.xlabel('Training examples')
plt.ylabel('Score')
plt.legend()
plt.show()
```

```
from sklearn.svm import SVC
▾
SVM_model = SVC(kernel='rbf', probability=True)   # rbf = default kernel
SVM_model.fit(X_train, y_train)
```

```
SVC
i
?
SVC(probability=True)
```

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
# Predict using SVM
y_pred = SVM_model.predict(X_test)
# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)
# Plot
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("SVM Confusion Matrix")
plt.show()
```

```
from sklearn.metrics import classification_report
# Predict using SVM
y_pred = SVM_model.predict(X_test)
# Print precision, recall, f1-score
print(classification_report(y_test, y_pred))
              precision    recall  f1-score   support
         0.0       0.77      0.82      0.79        99
         1.0       0.63      0.56      0.60        55
    accuracy                           0.73       154
```

```
       macro avg       0.70      0.69      0.70       154
    weighted avg       0.72      0.73      0.72       154
```

Close

```
If you run all this code like confusion matrix and ROC curve will be visuable
```

If you run all this code like confusion matrix and ROC curve will be visuable