Import Libraries

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
sns.set_theme(color_codes=True)
```

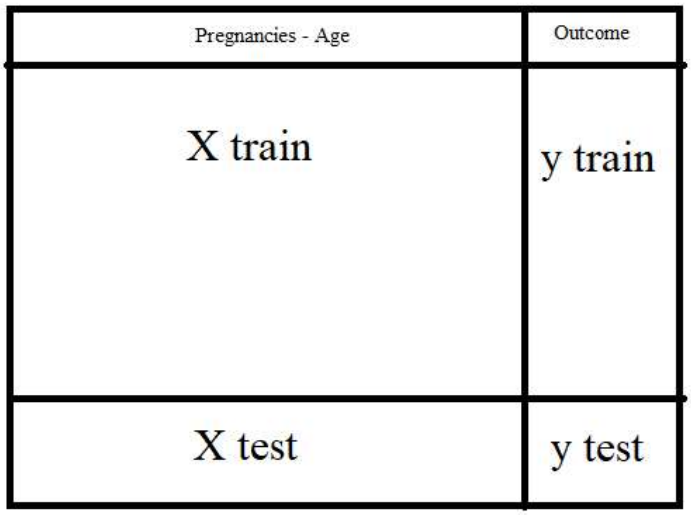
Classification Supervised Learning

You can get the dataset from here : Diabetes Dataset

```
df = pd.read_csv('diabetes.csv')
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigre
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	

Split the X and y then split the train and test data into like this :




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

```
#test size 10% and train size 90%
#Random state = Controls the randomness of the bootstrapping of the samples
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.1, random_state=0)
```

Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier(random_state=0) #Controls the randomness of the bootstrapping of the samples
dtree.fit(X_train, y_train)
```

```
DecisionTreeClassifier(random_state=0)
```

```
y_pred = dtree.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

```
Accuracy Score : 75.32 %
```

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
print('F-1 Score : ',(f1_score(y_test, y_pred)))
print('Precision Score : ',(precision_score(y_test, y_pred)))
print('Recall Score : ',(recall_score(y_test, y_pred)))
```

```
F-1 Score :  0.6666666666666667
Precision Score :  0.6129032258064516
Recall Score :  0.7307692307692307
```

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(random_state=0)
rfc.fit(X_train, y_train)
```

```
RandomForestClassifier(random_state=0)
```

```
y_pred = rfc.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

```
Accuracy Score : 80.52 %
```

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
print('F-1 Score : ',(f1_score(y_test, y_pred)))
print('Precision Score : ',(precision_score(y_test, y_pred)))
print('Recall Score : ',(recall_score(y_test, y_pred)))
```

```
F-1 Score :  0.7058823529411765
Precision Score :  0.72
Recall Score :  0.6923076923076923
```

Support Vector Machine Classifier

```
from sklearn import svm
svm = svm.SVC(random_state=0)
svm.fit(X_train, y_train)
```

```
SVC(random_state=0)
```

```
y_pred = svm.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

```
Accuracy Score : 85.71 %
```

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
print('F-1 Score : ',(f1_score(y_test, y_pred)))
print('Precision Score : ',(precision_score(y_test, y_pred)))
print('Recall Score : ',(recall_score(y_test, y_pred)))
```

```
F-1 Score :  0.7659574468085107
Precision Score :  0.8571428571428571
```

```
Recall Score :  0.6923076923076923
```

AdaBoost Classifier

```
from sklearn.ensemble import AdaBoostClassifier
ada = AdaBoostClassifier(random_state=0)
ada.fit(X_train, y_train)
```

```
AdaBoostClassifier
AdaBoostClassifier(random_state=0)
```

```
y_pred = ada.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

```
Accuracy Score : 79.22 %
```

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
print('F-1 Score : ',(f1_score(y_test, y_pred)))
print('Precision Score : ',(precision_score(y_test, y_pred)))
print('Recall Score : ',(recall_score(y_test, y_pred)))
```

```
F-1 Score :  0.7037037037037038
Precision Score :  0.6785714285714286
Recall Score :  0.7307692307692307
```

Logistic Regression Classifier

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train, y_train)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_logistic.py:458: Conv
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(
```

```
LogisticRegression
LogisticRegression()
```

```
y_pred = lr.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

```
Accuracy Score : 87.01 %
```

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
print('F-1 Score : ',(f1_score(y_test, y_pred)))
print('Precision Score : ',(precision_score(y_test, y_pred)))
print('Recall Score : ',(recall_score(y_test, y_pred)))
```

```
F-1 Score :  0.7916666666666666
Precision Score :  0.8636363636363636
Recall Score :  0.7307692307692307
```

Regression Supervised Learning

You can get the dataset from here : Mobile Price Dataset

```
df2 = pd.read_csv('Cellphone.csv')
df2.head()
```

	Product_id	Price	Sale	weight	resoloution	ppi	cpu core	cpu freq	internal mem	ram
0	203	2357	10	135.0	5.2	424	8	1.35	16.0	3.000
1	880	1749	10	125.0	4.0	233	2	1.30	4.0	1.000
2	40	1916	10	110.0	4.7	312	4	1.20	8.0	1.500
3	99	1315	11	118.5	4.0	233	2	1.30	4.0	0.512
4	880	1749	11	125.0	4.0	233	2	1.30	4.0	1.000

```
#Remove Product_id attribute because it's unnecessary
df2.drop(columns='Product_id', inplace = True)
df2.head()
```

	Price	Sale	weight	resoloution	ppi	cpu core	cpu freq	internal mem	ram	RearCam	Fro
0	2357	10	135.0	5.2	424	8	1.35	16.0	3.000	13.00	
1	1749	10	125.0	4.0	233	2	1.30	4.0	1.000	3.15	
2	1916	10	110.0	4.7	312	4	1.20	8.0	1.500	13.00	
3	1315	11	118.5	4.0	233	2	1.30	4.0	0.512	3.15	

Just like the Classification method, split the target and attribute then split the train and test data

```
X = df2.drop('Price', axis=1)
y = df2['Price']
```

```
#test size 20% and train size 80%
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2,random_state=0)
```

Decision Tree Regressor

```
from sklearn.tree import DecisionTreeRegressor
dtree = DecisionTreeRegressor(random_state=0)
dtree.fit(X_train, y_train)
```

```
DecisionTreeRegressor(random_state=0)
```

```
from sklearn import metrics
import math
y_pred = dtree.predict(X_test)
mae = metrics.mean_absolute_error(y_test, y_pred)
mse = metrics.mean_squared_error(y_test, y_pred)
r2 = metrics.r2_score(y_test, y_pred)
rmse = math.sqrt(mse)

print('MAE is {}'.format(mae))
print('MSE is {}'.format(mse))
print('R2 score is {}'.format(r2))
print('RMSE score is {}'.format(rmse))
```

```
MAE is 53.63636363636363
MSE is 19068.666666666668
R2 score is 0.9593394123291379
RMSE score is 138.0893430597259
```

Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(random_state=0)
rf.fit(X_train, y_train)
```

```
RandomForestRegressor(random_state=0)
```

```
from sklearn import metrics
import math
y_pred = rf.predict(X_test)
mae = metrics.mean_absolute_error(y_test, y_pred)
mse = metrics.mean_squared_error(y_test, y_pred)
r2 = metrics.r2_score(y_test, y_pred)
rmse = math.sqrt(mse)

print('MAE is {}'.format(mae))
print('MSE is {}'.format(mse))
print('R2 score is {}'.format(r2))
print('RMSE score is {}'.format(rmse))
```

```
MAE is 107.12181818181816
MSE is 17579.776721212114
R2 score is 0.962514209037145
RMSE score is 132.58875035692927
```

AdaBoost Regressor

```
from sklearn.ensemble import AdaBoostRegressor
ada = AdaBoostRegressor(random_state=0)
ada.fit(X_train, y_train)
```

```
AdaBoostRegressor(random_state=0)
```

```
from sklearn import metrics
import math
y_pred = ada.predict(X_test)
mae = metrics.mean_absolute_error(y_test, y_pred)
mse = metrics.mean_squared_error(y_test, y_pred)
r2 = metrics.r2_score(y_test, y_pred)
rmse = math.sqrt(mse)

print('MAE is {}'.format(mae))
print('MSE is {}'.format(mse))
print('R2 score is {}'.format(r2))
print('RMSE score is {}'.format(rmse))
```

```
MAE is 130.3148744244737
MSE is 24064.750340734758
R2 score is 0.9486861400373987
RMSE score is 155.1281739102693
```

✓ 0s completed at 1:57 AM