In [3]:
```python
import pandas as pd
```

In [4]:
```python
data = pd.read_csv('diabetesdataset.csv')
```

1. Display Top 5 Rows of The Dataset

In [5]:
```python
data.head()
```

Out[5]:

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

2. Check Last 5 Rows of The Dataset

In [6]:
```python
data.tail()
```

Out[6]:

	Unnamed: 0	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedi
763	763	10	101	76	48	180	32.9	
764	764	2	122	70	27	0	36.8	
765	765	5	121	72	23	112	26.2	
766	766	1	126	60	0	0	30.1	
767	767	1	93	70	31	0	30.4	

3. Find Shape of Our Dataset (Number of Rows And Number of Columns)

In [7]:
```python
data.shape
```

Out[7]:
```
(768, 10)
```

In [8]:

```
print("Number of Rows",data.shape[0])
print("Number of Columns",data.shape[1])
```

```
Number of Rows 768
Number of Columns 10
```

4. Get Information About Our Dataset Like Total Number Rows, Total Number of Columns, Datatypes of Each Column And Memory Requirement

In [9]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Unnamed: 0                768 non-null    int64  
 1   Pregnancies               768 non-null    int64  
 2   Glucose                   768 non-null    int64  
 3   BloodPressure             768 non-null    int64  
 4   SkinThickness             768 non-null    int64  
 5   Insulin                   768 non-null    int64  
 6   BMI                       768 non-null    float64
 7   DiabetesPedigreeFunction  768 non-null    float64
 8   Age                       768 non-null    int64  
 9   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(8)
memory usage: 60.1 KB
```

5. Check Null Values In The Dataset

In [10]:

```
data.isnull().sum()
```

Out[10]:

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

6. Get Overall Statistics About The Dataset

In [11]:
```
data.describe()
```

Out[11]:

	Unnamed: 0	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BI
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.0000
mean	383.500000	3.845052	120.894531	69.105469	20.536458	79.799479	31.9925
std	221.846794	3.369578	31.972618	19.355807	15.952218	115.244002	7.8841
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	191.750000	1.000000	99.000000	62.000000	0.000000	0.000000	27.3000
50%	383.500000	3.000000	117.000000	72.000000	23.000000	30.500000	32.0000
75%	575.250000	6.000000	140.250000	80.000000	32.000000	127.250000	36.6000
max	767.000000	17.000000	199.000000	122.000000	99.000000	846.000000	67.1000

In [12]:
```
import numpy as np
```

In [13]:
```
data_copy = data.copy(deep=True)
```

In [14]:
```
data.columns
```

Out[14]:
```
Index(['Unnamed: 0', 'Pregnancies', 'Glucose', 'BloodPressure',
       'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age',
       'Outcome'],
      dtype='object')
```

In [15]:
```
data_copy[['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI']] = data_copy[['Glucose', 'BloodPressure', 'SkinThickness', 'Insul
       'BMI']].replace(0,np.nan)
```

In [16]:
```
data_copy.isnull().sum()
```

Out[16]:
```
Unnamed: 0                    0
Pregnancies                   0
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
dtype: int64
```

In [17]:

```
data['Glucose'] = data['Glucose'].replace(0,data['Glucose'].mean())
data['BloodPressure'] = data['BloodPressure'].replace(0,data['BloodPressure'].m
data['SkinThickness'] = data['SkinThickness'].replace(0,data['SkinThickness'].m
data['Insulin'] = data['Insulin'].replace(0,data['Insulin'].mean())
data['BMI'] = data['BMI'].replace(0,data['BMI'].mean())
```

In [ ]:

7. Store Feature Matrix In X and Response(Target) In Vector y

In [18]:

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

In [ ]:

8. Splitting The Dataset Into The Training Set And Test Set

In [19]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,
                                               random_state=42)
```

9. Scikit-Learn Pipeline

In [20]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier

from sklearn.pipeline import Pipeline
```

In [21]:
```
pipeline_lr  = Pipeline([('scalar1',StandardScaler()),
                         ('lr_classifier',LogisticRegression())])

pipeline_knn = Pipeline([('scalar2',StandardScaler()),
                          ('knn_classifier',KNeighborsClassifier())])

pipeline_svc = Pipeline([('scalar3',StandardScaler()),
                         ('svc_classifier',SVC())])

pipeline_dt = Pipeline([('dt_classifier',DecisionTreeClassifier())])
pipeline_rf = Pipeline([('rf_classifier',RandomForestClassifier(max_depth=3))])
pipeline_gbc = Pipeline([('gbc_classifier',GradientBoostingClassifier())])
```

In [22]:
```
pipelines = [pipeline_lr,
            pipeline_knn,
            pipeline_svc,
            pipeline_dt,
            pipeline_rf,
            pipeline_gbc]
```

In [23]:
```
pipelines
```

Out[23]:
```
[Pipeline(steps=[('scalar1', StandardScaler()),
                 ('lr_classifier', LogisticRegression())]),
 Pipeline(steps=[('scalar2', StandardScaler()),
                 ('knn_classifier', KNeighborsClassifier())]),
 Pipeline(steps=[('scalar3', StandardScaler()), ('svc_classifier', SVC())]),
 Pipeline(steps=[('dt_classifier', DecisionTreeClassifier())]),
 Pipeline(steps=[('rf_classifier', RandomForestClassifier(max_depth=3))]),
 Pipeline(steps=[('gbc_classifier', GradientBoostingClassifier())])]
```

In [24]:
```
for pipe in pipelines:
    pipe.fit(X_train,y_train)
```

In [25]:
```
pipe_dict = {0:'LR',
             1:'KNN',
             2:'SVC',
             3:'DT',
             4: 'RF',
             5: 'GBC'}
```

In [26]:
```
pipe_dict
```

Out[26]:
```
{0: 'LR', 1: 'KNN', 2: 'SVC', 3: 'DT', 4: 'RF', 5: 'GBC'}
```

In [27]:

```
for i,model in enumerate(pipelines):
    print("{} Test Accuracy:{}".format(pipe_dict[i],model.score(X_test,y_test)*
```

```
LR Test Accuracy:76.62337662337663
KNN Test Accuracy:68.83116883116884
SVC Test Accuracy:73.37662337662337
DT Test Accuracy:72.72727272727273
RF Test Accuracy:75.97402597402598
GBC Test Accuracy:75.97402597402598
```

In [28]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [29]:

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

In [30]:

```
rf =RandomForestClassifier(max_depth=3)
```

In [31]:

```
rf.fit(X,y)
```

Out[31]:

```
▾       RandomForestClassifier

RandomForestClassifier(max_depth=3)
```

Prediction on New Data

In [32]:

```
new_data = pd.DataFrame({
    'Pregnancies':6,
    'Glucose':148.0,
    'BloodPressure':72.0,
    'SkinThickness':35.0,
    'Insulin':79.799479,
    'BMI':33.6,
    'DiabetesPedigreeFunction':0.627,
    'Age':50,
},index=[0])
```

```
In [33]: p = rf.predict(new_data)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[33], line 1
----> 1 p = rf.predict(new_data)

File D:\anaconda\Lib\site-packages\sklearn\ensemble\_forest.py:823, in ForestClassifier.predict(self, X)
    802 def predict(self, X):
    803     """
    804     Predict class for X.
    805 
   (...)
    821         The predicted classes.
    822     """
--> 823     proba = self.predict_proba(X)
    825     if self.n_outputs_ == 1:
    826         return self.classes_.take(np.argmax(proba, axis=1), axis=0)

File D:\anaconda\Lib\site-packages\sklearn\ensemble\_forest.py:865, in ForestClassifier.predict_proba(self, X)
    863 check_is_fitted(self)
    864 # Check data
--> 865 X = self._validate_X_predict(X)
    867 # Assign chunk of trees to jobs
    868 n_jobs, _, _ = _partition_estimators(self.n_estimators, self.n_jobs)

File D:\anaconda\Lib\site-packages\sklearn\ensemble\_forest.py:599, in BaseForest._validate_X_predict(self, X)
    596 """
    597 Validate X whenever one tries to predict, apply, predict_proba."""
    598 check_is_fitted(self)
--> 599 X = self._validate_data(X, dtype=DTYPE, accept_sparse="csr", reset=False)
    600 if issparse(X) and (X.indices.dtype != np.intc or X.indptr.dtype != np.intc):
    601     raise ValueError("No support for np.int64 index based sparse matrices")

File D:\anaconda\Lib\site-packages\sklearn\base.py:579, in BaseEstimator._validate_data(self, X, y, reset, validate_separately, cast_to_ndarray, **check_params)
    508 def _validate_data(
    509     self,
    510     X="no_validation",
   (...)
    515     **check_params,
    516 ):
    517     """Validate input data and set or check the `n_features_in_` attribute.
    518 
    519     Parameters
   (...)
    577         validated.
    578     """
--> 579     self._check_feature_names(X, reset=reset)
    581     if y is None and self._get_tags()["requires_y"]:
    582         raise ValueError(
    583             f"This {self.__class__.__name__} estimator "
    584             "requires y to be passed, but the target y is None."
    585         )

File D:\anaconda\Lib\site-packages\sklearn\base.py:506, in BaseEstimator._che
ck_feature_names(self, X, reset)
    501 if not missing_names and not unexpected_names:
    502     message += (
    503         "Feature names must be in the same order as they were in fi
t.\n"
    504     )
--> 506 raise ValueError(message)

ValueError: The feature names should match those that were passed during fit.
Feature names seen at fit time, yet now missing:
- Unnamed: 0
```

In [34]:

```
if p[0] == 0:
    print('non-diabetic')
else:
    print('diabetic')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[34], line 1
----> 1 if p[0] == 0:
      2     print('non-diabetic')
      3 else:

NameError: name 'p' is not defined
```

18. Save Model Using Joblib

In [35]:

```
import joblib
```

In [36]:

```
joblib.dump(rf,'model_joblib_diabetes')
```

Out[36]:

```
['model_joblib_diabetes']
```

In [37]:

```
model = joblib.load('model_joblib_diabetes')
```

In [38]:

```
model.predict(new_data)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[38], line 1
----> 1 model.predict(new_data)

File D:\anaconda\Lib\site-packages\sklearn\ensemble\_forest.py:823, in ForestClassifier.predict(self, X)
    802 def predict(self, X):
    803     """
    804     Predict class for X.
    805 
   (...)
    821         The predicted classes.
    822     """
--> 823     proba = self.predict_proba(X)
    825     if self.n_outputs_ == 1:
    826         return self.classes_.take(np.argmax(proba, axis=1), axis=0)

File D:\anaconda\Lib\site-packages\sklearn\ensemble\_forest.py:865, in ForestClassifier.predict_proba(self, X)
    863 check_is_fitted(self)
    864 # Check data
--> 865 X = self._validate_X_predict(X)
    867 # Assign chunk of trees to jobs
    868 n_jobs, _, _ = _partition_estimators(self.n_estimators, self.n_jobs)

File D:\anaconda\Lib\site-packages\sklearn\ensemble\_forest.py:599, in BaseForest._validate_X_predict(self, X)
    596 """
    597 Validate X whenever one tries to predict, apply, predict_proba."""
    598 check_is_fitted(self)
--> 599 X = self._validate_data(X, dtype=DTYPE, accept_sparse="csr", reset=False)
    600 if issparse(X) and (X.indices.dtype != np.intc or X.indptr.dtype != np.intc):
    601     raise ValueError("No support for np.int64 index based sparse matrices")

File D:\anaconda\Lib\site-packages\sklearn\base.py:579, in BaseEstimator._validate_data(self, X, y, reset, validate_separately, cast_to_ndarray, **check_params)
    508 def _validate_data(
    509     self,
    510     X="no_validation",
   (...)
    515     **check_params,
    516 ):
    517     """Validate input data and set or check the `n_features_in_` attribute.
    518 
    519     Parameters
   (...)
    577         validated.
    578     """
--> 579     self._check_feature_names(X, reset=reset)
    581     if y is None and self._get_tags()["requires_y"]:
    582         raise ValueError(
    583             f"This {self.__class__.__name__} estimator "
    584             "requires y to be passed, but the target y is None."
    585         )

File D:\anaconda\Lib\site-packages\sklearn\base.py:506, in BaseEstimator._che
ck_feature_names(self, X, reset)
    501 if not missing_names and not unexpected_names:
    502     message += (
    503         "Feature names must be in the same order as they were in fi
t.\n"
    504     )
--> 506 raise ValueError(message)

ValueError: The feature names should match those that were passed during fit.
Feature names seen at fit time, yet now missing:
- Unnamed: 0
```

GUI

In [ ]:

```

```

In [39]:

```
from tkinter import *
import joblib
```

In [40]:

```
from tkinter import *
import joblib
import numpy as np
from sklearn import *
def show_entry_fields():
    p1=float(e1.get())
    p2=float(e2.get())
    p3=float(e3.get())
    p4=float(e4.get())
    p5=float(e5.get())
    p6=float(e6.get())
    p7=float(e7.get())
    p8=float(e8.get())

    model = joblib.load('model_joblib_diabetes')
    result=model.predict([[p1,p2,p3,p4,p5,p6,p7,p8]])

    if result == 0:
        Label(master, text="Non-Diabetic").grid(row=31)
    else:
        Label(master, text="Diabetic").grid(row=31)


master = Tk()
master.title("Diabetes Prediction Using Machine Learning")


label = Label(master, text = "Diabetes Prediction Using Machine Learning"
                          , bg = "black", fg = "white"). \
                               grid(row=0,columnspan=2)


Label(master, text="Pregnancies").grid(row=1)
Label(master, text="Glucose").grid(row=2)
Label(master, text="Enter Value of BloodPressure").grid(row=3)
Label(master, text="Enter Value of SkinThickness").grid(row=4)
Label(master, text="Enter Value of Insulin").grid(row=5)
Label(master, text="Enter Value of BMI").grid(row=6)
Label(master, text="Enter Value of DiabetesPedigreeFunction").grid(row=7)
Label(master, text="Enter Value of Age").grid(row=8)


e1 = Entry(master)
e2 = Entry(master)
e3 = Entry(master)
e4 = Entry(master)
e5 = Entry(master)
e6 = Entry(master)
e7 = Entry(master)
e8 = Entry(master)


e1.grid(row=1, column=1)
e2.grid(row=2, column=1)
e3.grid(row=3, column=1)
e4.grid(row=4, column=1)
e5.grid(row=5, column=1)
```

```
e6.grid(row=6, column=1)
e7.grid(row=7, column=1)
e8.grid(row=8, column=1)


Button(master, text='Predict', command=show_entry_fields).grid()

mainloop()
```

```
D:\anaconda\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not ha
ve valid feature names, but RandomForestClassifier was fitted with feature na
mes
  warnings.warn(
Exception in Tkinter callback
Traceback (most recent call last):
  File "D:\anaconda\Lib\tkinter\__init__.py", line 1948, in __call__
    return self.func(*args)
           ^^^^^^^^^^^^^^^^
  File "C:\Users\Manic\AppData\Local\Temp\ipykernel_19668\669078951.py", line
16, in show_entry_fields
    result=model.predict([[p1,p2,p3,p4,p5,p6,p7,p8]])
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\anaconda\Lib\site-packages\sklearn\ensemble\_forest.py", line 823,
in predict
    proba = self.predict_proba(X)
            ^^^^^^^^^^^^^^^^^^^^^
  File "D:\anaconda\Lib\site-packages\sklearn\ensemble\_forest.py", line 865,
in predict_proba
    X = self._validate_X_predict(X)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\anaconda\Lib\site-packages\sklearn\ensemble\_forest.py", line 599,
in _validate_X_predict
    X = self._validate_data(X, dtype=DTYPE, accept_sparse="csr", reset=False)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\anaconda\Lib\site-packages\sklearn\base.py", line 625, in _validat
e_data
    self._check_n_features(X, reset=reset)
  File "D:\anaconda\Lib\site-packages\sklearn\base.py", line 414, in _check_n
_features
    raise ValueError(
ValueError: X has 8 features, but RandomForestClassifier is expecting 9 featu
res as input.
Exception in Tkinter callback
Traceback (most recent call last):
  File "D:\anaconda\Lib\tkinter\__init__.py", line 1948, in __call__
    return self.func(*args)
           ^^^^^^^^^^^^^^^^
  File "C:\Users\Manic\AppData\Local\Temp\ipykernel_19668\669078951.py", line
6, in show_entry_fields
    p1=float(e1.get())
       ^^^^^^^^^^^^^^^
ValueError: could not convert string to float: '34%'
D:\anaconda\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not ha
ve valid feature names, but RandomForestClassifier was fitted with feature na
mes
  warnings.warn(
Exception in Tkinter callback
Traceback (most recent call last):
  File "D:\anaconda\Lib\tkinter\__init__.py", line 1948, in __call__
    return self.func(*args)
           ^^^^^^^^^^^^^^^^
  File "C:\Users\Manic\AppData\Local\Temp\ipykernel_19668\669078951.py", line
16, in show_entry_fields
    result=model.predict([[p1,p2,p3,p4,p5,p6,p7,p8]])
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\anaconda\Lib\site-packages\sklearn\ensemble\_forest.py", line 823,
in predict
    proba = self.predict_proba(X)
            ^^^^^^^^^^^^^^^^^^^^^
  File "D:\anaconda\Lib\site-packages\sklearn\ensemble\_forest.py", line 865,
in predict_proba
    X = self._validate_X_predict(X)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\anaconda\Lib\site-packages\sklearn\ensemble\_forest.py", line 599,
in _validate_X_predict
    X = self._validate_data(X, dtype=DTYPE, accept_sparse="csr", reset=False)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\anaconda\Lib\site-packages\sklearn\base.py", line 625, in _validat
e_data
    self._check_n_features(X, reset=reset)
  File "D:\anaconda\Lib\site-packages\sklearn\base.py", line 414, in _check_n
_features
    raise ValueError(
ValueError: X has 8 features, but RandomForestClassifier is expecting 9 featu
res as input.
```

In [ ]: