

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
path = "../CSV/diabetes.csv"
```

In [3]:

```
df = pd.read_csv(path)
```

In [46]:

```
df.head()
```

Out[46]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28

In [48]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
Pregnancies      768 non-null int64
Glucose          768 non-null int64
BloodPressure    768 non-null int64
SkinThickness    768 non-null int64
Insulin          768 non-null int64
BMI              768 non-null float64
DiabetesPedigreeFunction  768 non-null float64
Age              768 non-null int64
Outcome          768 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [49]:

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

Out[49]:

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

In [51]:

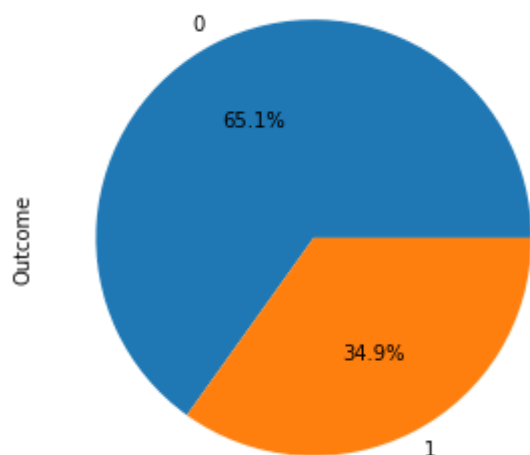
```
pd.set_option('display.float_format', '{:.2f}'.format)
df.describe()
```

Out[51]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeF
count	768.00	768.00	768.00	768.00	768.00	768.00	
mean	3.85	120.89	69.11	20.54	79.80	31.99	
std	3.37	31.97	19.36	15.95	115.24	7.88	
min	0.00	0.00	0.00	0.00	0.00	0.00	
25%	1.00	99.00	62.00	0.00	0.00	27.30	
50%	3.00	117.00	72.00	23.00	30.50	32.00	
75%	6.00	140.25	80.00	32.00	127.25	36.60	
max	17.00	199.00	122.00	99.00	846.00	67.10	

In [8]:

```
plt.figure(figsize=(5,5))
df["Outcome"].value_counts().plot(kind="pie", autopct="%1.1f%%")
plt.show()
```



In [5]:

```
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

In [6]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

In [7]:

```
def create_model(model):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(classification_report(y_test, y_pred))
    return model
```

Naive Aggregation

In [12]:

```
lr = LogisticRegression()
dt1 = DecisionTreeClassifier()
dt2 = DecisionTreeClassifier(criterion="entropy")
```

In [13]:

```
model_list = [('lr', lr), ('dt1', dt1), ('dt2', dt2)]
```

In [14]:

```
from sklearn.ensemble import VotingClassifier
```

1) Hard votina

In [17]:

```
vc_hard = VotingClassifier(estimators=model_list)
```

In [19]:

```
create_model(vc_hard)
```

	precision	recall	f1-score	support
0	0.78	0.86	0.82	146
1	0.71	0.59	0.65	85
accuracy			0.76	231
macro avg	0.75	0.73	0.73	231
weighted avg	0.76	0.76	0.76	231

Out[19]:

```
VotingClassifier(estimators=[('lr',
                             LogisticRegression(C=1.0, class_weight=None,
                                                  dual=False, fit_intercept=True,
                                                  intercept_scaling=1,
                                                  l1_ratio=None, max_iter=1000,
                                                  multi_class='warn',
                                                  n_jobs=None, penalty='l2',
                                                  random_state=None,
                                                  solver='warn', tol=0.0001,
                                                  verbose=0, warm_start=False),
                             ('dt1',
                              DecisionTreeClassifier(class_weight=None,
                                                       criterion='gini',
                                                       max_depth=None,
                                                       max_features=None,
                                                       min_impurity_decrease=0.0,
                                                       min_impurity_split=None,
                                                       min_samples_leaf=1,
                                                       min_samples_split=2,
                                                       min_weight_fraction_leaf=0.0,
                                                       presort=False,
                                                       random_state=None,
                                                       splitter='best'))],
                 flatten_transform=True, n_jobs=None, voting='hard',
                 weights=None)
```

2) Soft voting

In [21]:

```
vc_soft = VotingClassifier(estimators=model_list,voting='soft')
```

In [22]:

```
create_model(vc_soft)
```

	precision	recall	f1-score	support
0	0.77	0.86	0.81	146
1	0.69	0.55	0.61	85
accuracy			0.74	231
macro avg	0.73	0.70	0.71	231
weighted avg	0.74	0.74	0.74	231

Out[22]:

```
VotingClassifier(estimators=[('lr',
                             LogisticRegression(C=1.0, class_weight=None,
                                                  dual=False, fit_intercept=True,
                                                  intercept_scaling=1,
                                                  l1_ratio=None, max_iter=100,
                                                  multi_class='warn',
                                                  n_jobs=None, penalty='l2',
                                                  random_state=None,
                                                  solver='warn', tol=0.0001,
                                                  verbose=0, warm_start=False),
                             ('dt1',
                              DecisionTreeClassifier(class_weight=None,
                                                       criterion='gini',
                                                       max_depth=None,
                                                       max_features=None,
                                                       max_leaf_nodes=None,
                                                       min_impurity_decrease=0.0,
                                                       min_impurity_split=None,
                                                       min_samples_leaf=1,
                                                       min_samples_split=2,
                                                       min_weight_fraction_leaf=0.0,
                                                       presort=False,
                                                       random_state=None,
                                                       splitter='best'))],
                  flatten_transform=True, n_jobs=None, voting='soft',
                  weights=None)
```

Bootstrap Aggregation

In [12]:

```
from sklearn.ensemble import BaggingClassifier
```

1) Bagging

In [16]:

```
# use odd number of n_estimators for better output
bg1 = BaggingClassifier(LogisticRegression(),n_estimators=11,random_state=1)
```

In [17]:

```
create_model(bg1)
```

	precision	recall	f1-score	support
0	0.78	0.90	0.83	146
1	0.76	0.55	0.64	85
accuracy			0.77	231
macro avg	0.77	0.73	0.74	231
weighted avg	0.77	0.77	0.76	231

Out[17]:

```
BaggingClassifier(base_estimator=LogisticRegression(C=1.0, class_weight=None,
dual=False,
fit_intercept=True,
intercept_scaling=1,
l1_ratio=None, max_iter=
100,
multi_class='warn',
n_jobs=None, penalty='l
2',
random_state=None,
solver='warn', tol=0.000
1,
verbose=0,
warm_start=False),
bootstrap=True, bootstrap_features=False, max_features=1.
0,
max_samples=1.0, n_estimators=11, n_jobs=None,
oob_score=False, random_state=1, verbose=0, warm_start=False)
se)
```

In [34]:

```
bg2 = BaggingClassifier(DecisionTreeClassifier(),n_estimators=900,random_state=1)
```

In [35]:

```
create_model(bg2)
```

	precision	recall	f1-score	support
0	0.81	0.88	0.84	146
1	0.75	0.64	0.69	85
accuracy			0.79	231
macro avg	0.78	0.76	0.76	231
weighted avg	0.78	0.79	0.78	231

Out[35]:

```
BaggingClassifier(base_estimator=DecisionTreeClassifier(class_weight=None,
                                                         criterion='gini',
                                                         max_depth=None,
                                                         max_features=None,
                                                         max_leaf_nodes=None,
                                                         min_impurity_decreas
e=0.0,
                                                         min_impurity_split=N
one,
                                                         min_samples_leaf=1,
                                                         min_samples_split=2,
                                                         min_weight_fraction_
leaf=0.0,
                                                         presort=False,
                                                         random_state=None,
                                                         splitter='best'),
                 bootstrap=True, bootstrap_features=False, max_features=1.
0,
                 max_samples=1.0, n_estimators=900, n_jobs=None,
                 oob_score=False, random_state=1, verbose=0, warm_start=Fal
se)
```

2) Pasting

In [35]:

```
bg3 = BaggingClassifier(LogisticRegression(),n_estimators=7,random_state=1,bootstrap=False)
```

In [36]:

```
create_model(bg3)
```

	precision	recall	f1-score	support
0	0.78	0.90	0.84	146
1	0.77	0.55	0.64	85
accuracy			0.77	231
macro avg	0.77	0.73	0.74	231
weighted avg	0.77	0.77	0.76	231

Out[36]:

```
BaggingClassifier(base_estimator=LogisticRegression(C=1.0, class_weight=None,
dual=False,
fit_intercept=True,
intercept_scaling=1,
l1_ratio=None, max_iter=
100,
multi_class='warn',
n_jobs=None, penalty='l
2',
random_state=None,
solver='warn', tol=0.000
1,
verbose=0,
warm_start=False),
bootstrap=False, bootstrap_features=False, max_features=1.
0,
max_samples=1.0, n_estimators=7, n_jobs=None, oob_score=False,
lse,
random_state=1, verbose=0, warm_start=False)
```

Random Forest

In [37]:

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

In [44]:

```
rf1 = RandomForestClassifier(n_estimators=13)
```


In [45]:

```
create_model(rf1)
```

	precision	recall	f1-score	support
0	0.79	0.87	0.83	146
1	0.73	0.60	0.66	85
accuracy			0.77	231
macro avg	0.76	0.73	0.74	231
weighted avg	0.77	0.77	0.77	231

Out[45]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                        max_depth=None, max_features='auto', max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, n_estimators=13,
                        n_jobs=None, oob_score=False, random_state=None,
                        verbose=0, warm_start=False)
```

In [150]:

```
rf2 = RandomForestClassifier(n_estimators=13,max_features=5,random_state=1)
```

In [151]:

```
create_model(rf2)
```

	precision	recall	f1-score	support
0	0.81	0.84	0.83	146
1	0.71	0.66	0.68	85
accuracy			0.77	231
macro avg	0.76	0.75	0.75	231
weighted avg	0.77	0.77	0.77	231

Out[151]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                        max_depth=None, max_features=5, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, n_estimators=13,
                        n_jobs=None, oob_score=False, random_state=1, verbose
                        =0,
                        warm_start=False)
```

In [180]:

```
rf3 = RandomForestClassifier(n_estimators=13,max_depth=15,random_state=1)
```

In [181]:

```
create_model(rf3)
```

	precision	recall	f1-score	support
0	0.84	0.87	0.85	146
1	0.76	0.71	0.73	85
accuracy			0.81	231
macro avg	0.80	0.79	0.79	231
weighted avg	0.81	0.81	0.81	231

Out[181]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                        max_depth=15, max_features='auto', max_leaf_nodes=None,
e,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, n_estimators=13,
                        n_jobs=None, oob_score=False, random_state=1, verbose
=0,
                        warm_start=False)
```

Stacking

In [182]:

```
from mlxtend.classifier import StackingClassifier
```

In [183]:

```
lr = LogisticRegression()
dt1 = DecisionTreeClassifier()
dt2 = DecisionTreeClassifier(criterion="entropy")
model_list = [lr,dt1,dt2]
```

In [184]:

```
# meta classifier model --> Logistic Reg / Decision Tree
meta_lr = LogisticRegression()
```

In [185]:

```
stack1 = StackingClassifier(classifiers=model_list,meta_classifier=meta_lr)
```

In [186]:

```
create_model(stack1)
```

	precision	recall	f1-score	support
0	0.78	0.86	0.82	146
1	0.70	0.59	0.64	85
accuracy			0.76	231
macro avg	0.74	0.72	0.73	231
weighted avg	0.75	0.76	0.75	231

Out[186]:

```
StackingClassifier(average_probab=False,
                    classifiers=[LogisticRegression(C=1.0, class_weight=None,
                                                    dual=False,
                                                    fit_intercept=True,
                                                    intercept_scaling=1,
                                                    l1_ratio=None, max_iter=1
00,
                                                    multi_class='warn',
                                                    n_jobs=None, penalty='l
2',
                                                    random_state=None,
                                                    solver='warn', tol=0.000
1,
                                                    verbose=0,
                                                    warm_start=False),
                                DecisionTreeClassifier(class_weight=None,
                                                        criterion='gini',
                                                        max...
                    meta_classifier=LogisticRegression(C=1.0, class_weight=No
ne,
                                                    dual=False,
                                                    fit_intercept=True,
                                                    intercept_scaling=1,
                                                    l1_ratio=None,
                                                    max_iter=100,
                                                    multi_class='warn',
                                                    n_jobs=None, penalty
='l2',
                                                    random_state=None,
                                                    solver='warn', tol=0.0
001,
                                                    verbose=0,
                                                    warm_start=False),
                    store_train_meta_features=False, use_clones=True,
                    use_features_in_secondary=False, use_probab=False,
                    verbose=0)
```

In [188]:

```
stack1.meta_clf_.coef_
```

Out[188]:

```
array([[0.46143284, 3.75889513, 3.75889513]])
```

In [189]:

```
meta_dt = DecisionTreeClassifier()
```

In [190]:

```
stack2 = StackingClassifier(classifiers=model_list,meta_classifier=meta_dt)
```

In [191]:

```
create_model(stack2)
```

	precision	recall	f1-score	support
0	0.73	0.77	0.75	146
1	0.57	0.52	0.54	85
accuracy			0.68	231
macro avg	0.65	0.65	0.65	231
weighted avg	0.67	0.68	0.68	231

Out[191]:

```
StackingClassifier(average_probab=False,
                    classifiers=[LogisticRegression(C=1.0, class_weight=None,
                                                    dual=False,
                                                    fit_intercept=True,
                                                    intercept_scaling=1,
                                                    l1_ratio=None, max_iter=1
00,
                                                    multi_class='warn',
                                                    n_jobs=None, penalty='l
2',
                                                    random_state=None,
                                                    solver='warn', tol=0.000
1,
                                                    verbose=0,
                                                    warm_start=False),
                                DecisionTreeClassifier(class_weight=None,
                                                        criterion='gini',
                                                        max_...
meta_classifier=DecisionTreeClassifier(class_weight=None,
                                        criterion='gini',
                                        max_depth=None,
                                        max_features=None,
                                        max_leaf_nodes=Non
e,
                                        min_impurity_decre
ase=0.0,
                                        min_impurity_split
=None,
                                        min_samples_leaf=
1,
                                        min_samples_split=
2,
                                        min_weight_fractio
n_leaf=0.0,
                                        presort=False,
                                        random_state=None,
                                        splitter='best'),
store_train_meta_features=False, use_clones=True,
use_features_in_secondary=False, use_probab=False,
verbose=0)
```

In [193]:

```
stack2.meta_clf_.feature_importances_
```

Out[193]:

```
array([0., 1., 0.])
```

In []: