Boosting

LECTURE FLOW

- Boosting
- Different Types:
  - AdaBoost
  - GradientBoost
  - XGBoost
- Perform boosting on diabetes
- Tune the hyper parameter and get better models
- Compare the models

Contents:

Adaboost

Link: https://www.youtube.com/watch?v=LsK-xG1cLYA&t=582s (https://www.youtube.com/watch?v=LsK-xG1cLYA&t=582s)

In [8]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [9]:
```
db = pd.read_csv('diabetes.csv')
```

In [10]:
```
db
```

Out[10]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

In [15]:
```
# Splitting into target and features
y = db['Outcome']
X = db.drop(['Outcome'], axis = 1)

# Convert X into standard form, standardizing all features
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)
```

In [16]:
```
x_scaled
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
C:\Users\URVISH~1\AppData\Local\Temp/ipykernel_2636/471005823.py in <module>
----> 1 x_scaled

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

In [17]:
```
# train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size = 0.2, random_state = 10)
```

In [18]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[18]:
```
((614, 8), (154, 8), (614,), (154,))
```

Building AdaBoostClassifier

class sklearn.ensemble.AdaBoostClassifier(base_estimator=None, *, n_estimators=50, learning_rate=1.0, algorithm='SAMME.R', random_state=None)

In [19]:
```
from sklearn.ensemble import AdaBoostClassifier

abc = AdaBoostClassifier(random_state = 10) # all other parameters kept default

abc.fit(X_train, y_train)
```

Out[19]:
```
AdaBoostClassifier(random_state=10)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [20]:
```
# Performance of the model
from sklearn.metrics import classification_report
report = classification_report(y_test, abc.predict(X_test))
print('REPORT:\n', report)
```

```
REPORT:
               precision    recall  f1-score   support

           0       0.74      0.88      0.81        95
           1       0.73      0.51      0.60        59

    accuracy                           0.74       154
   macro avg       0.74      0.70      0.70       154
weighted avg       0.74      0.74      0.73       154
```

Improving the model using hyperparameter tuning

In [21]:
```
from sklearn.model_selection import GridSearchCV

abc_gs = GridSearchCV(abc, {'n_estimators':range(25, 75), # n_estimators - default value is 50; 75-25
                            'learning_rate':[0, 0.25, 0.5, 0.75, 1]}) # learning_rate: weightage of each classifier, value betweer
```

In [22]:
```
abc_gs.fit(X_train, y_train)
```

```
C:\Users\Urvi Sharma\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:378: FitFailedWarning:
250 fits failed out of a total of 1250.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
250 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\Urvi Sharma\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 686, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\Urvi Sharma\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py", line 124, in fit
    self._validate_params()
  File "C:\Users\Urvi Sharma\anaconda3\lib\site-packages\sklearn\base.py", line 570, in _validate_params
    validate_parameter_constraints(
  File "C:\Users\Urvi Sharma\anaconda3\lib\site-packages\sklearn\utils\_param_validation.py", line 97, in validate_parameter_co
nstraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'learning_rate' parameter of AdaBoostClassifier must be a float in t
he range (0, inf). Got 0 instead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\Urvi Sharma\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:953: UserWarning: One or more of the test s
cores are non-finite: [       nan        nan        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan 0.76061575 0.76386779 0.76060243 0.76222844
 0.76222844 0.7605891  0.75896308 0.76221511 0.76057577 0.76221511
 0.75732374 0.75732374 0.7605891  0.76060243 0.76060243 0.76060243
 0.76060243 0.76385446 0.76222844 0.76874583 0.76874583 0.76873251
 0.76385446 0.76385446 0.76548047 0.76710649 0.77199787 0.77199787
 0.77362388 0.77362388 0.77199787 0.77037185 0.77199787 0.77037185
 0.77037185 0.77037185 0.76874583 0.77199787 0.77199787 0.77199787
 0.77199787 0.77037185 0.77037185 0.77037185 0.77199787 0.77199787
 0.76873251 0.77035852 0.77035852 0.77198454 0.75572438 0.75245902
 0.75408503 0.75896308 0.75735039 0.76385446 0.76548047 0.76060243
 0.76548047 0.76385446 0.76221511 0.76546715 0.7605891  0.76221511
 0.76384113 0.75733707 0.75733707 0.75733707 0.75571105 0.76221511
 0.75896308 0.7605891  0.75896308 0.75733707 0.75408503 0.75408503
 0.75408503 0.750833   0.75408503 0.75081967 0.75244569 0.75733707
 0.75733707 0.75896308 0.75733707 0.75571105 0.75733707 0.75735039
 0.76060243 0.7654938  0.7654938  0.7654938  0.76386779 0.7654938
 0.7622551  0.76062908 0.75572438 0.75735039 0.76064241 0.76389444
 0.75405838 0.75080634 0.75732374 0.75733707 0.75569772 0.75732374
 0.75081967 0.7459283  0.75245902 0.75081967 0.74268959 0.75245902
 0.74756764 0.75247234 0.75897641 0.75572438 0.75572438 0.75732374
 0.75568439 0.75894975 0.75569772 0.75571105 0.75408503 0.75571105
 0.75408503 0.75733707 0.75896308 0.7540717  0.75405838 0.75893643
 0.75404505 0.7605891  0.75733707 0.76221511 0.7605891  0.75732374
 0.7540717  0.75569772 0.75077969 0.75241903 0.74752766 0.75243236
 0.75569772 0.7540717  0.7540717  0.75408503 0.75572438 0.76060243
 0.75572438 0.76060243 0.75564441 0.74588831 0.75889644 0.7621618
 0.75728375 0.75728375 0.75569772 0.75569772 0.74752766 0.7540717
 0.75243236 0.75405838 0.75079302 0.75894975 0.75731041 0.75569772
 0.75735039 0.75569772 0.75241903 0.75079302 0.75404505 0.74427562
 0.74430228 0.74755431 0.74755431 0.74918033 0.74755431 0.74590164
 0.75080634 0.74266293 0.75245902 0.74594162 0.75247234 0.749167
 0.74591497 0.74754098 0.75243236 0.75732374 0.75080634 0.75896308
 0.75572438 0.75569772 0.75080634 0.75731041 0.75077969 0.749167
 0.7524057  0.749167   0.74427562 0.74752766]
  warnings.warn(
```

Out[22]:
```
GridSearchCV(estimator=AdaBoostClassifier(random_state=10),
             param_grid={'learning_rate': [0, 0.25, 0.5, 0.75, 1],
                         'n_estimators': range(25, 75)})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [23]:
```
abc_gs.best_params_
```

Out[23]:
```
{'learning_rate': 0.25, 'n_estimators': 53}
```

In [24]:
```
## Building the best model

ada_best = AdaBoostClassifier(learning_rate = 0.25, n_estimators = 53, random_state = 10)
ada_best.fit(X_train, y_train)
```

Out[24]:
```
AdaBoostClassifier(learning_rate=0.25, n_estimators=53, random_state=10)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [25]:
```
report = classification_report(y_test, ada_best.predict(X_test))

print('REPORT:\n', report)
```

```
REPORT:
               precision    recall  f1-score   support

           0       0.75      0.94      0.84        95
           1       0.83      0.51      0.63        59

    accuracy                           0.77       154
   macro avg       0.79      0.72      0.73       154
weighted avg       0.78      0.77      0.76       154
```

Building - Gradient Boosting Classifier

Also involves decision tree, no specificaition for max depth of decision tree to be one, by default it is 3

In [26]:
```
from sklearn.ensemble import GradientBoostingClassifier

# creating an object of the classifier
gbc = GradientBoostingClassifier(random_state = 10)
gbc.fit(X_train, y_train)
```

Out[26]:
```
GradientBoostingClassifier(random_state=10)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [27]:
```
report = classification_report(y_test, gbc.predict(X_test))

print('REPORT:\n', report)
```

```
REPORT:
               precision    recall  f1-score   support

           0       0.79      0.89      0.84        95
           1       0.78      0.61      0.69        59

    accuracy                           0.79       154
   macro avg       0.78      0.75      0.76       154
weighted avg       0.79      0.79      0.78       154
```

Improving the model using hyper-parameter tuning

In [28]:
```
gbc_gs = GridSearchCV(gbc, {'n_estimators':range(75, 125),
                            'max_depth':range(1,5)})
gbc_gs.fit(X_train, y_train)
```

Out[28]:
```
GridSearchCV(estimator=GradientBoostingClassifier(random_state=10),
             param_grid={'max_depth': range(1, 5),
                         'n_estimators': range(75, 125)})
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [29]:
```
gbc_gs.best_params_
```

Out[29]:
```
{'max_depth': 2, 'n_estimators': 76}
```

Best Gradient Boosting Model

In [30]:
```
grad_best = GradientBoostingClassifier(max_depth = 2, n_estimators = 76, random_state = 10)
grad_best.fit(X_train, y_train)
```

Out[30]:
```
GradientBoostingClassifier(max_depth=2, n_estimators=76, random_state=10)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [31]:
```
report = classification_report(y_test, grad_best.predict(X_test))

print('REPORT:\n', report)
```

```
REPORT:
               precision    recall  f1-score   support

           0       0.75      0.92      0.82        95
           1       0.79      0.51      0.62        59

    accuracy                           0.76       154
   macro avg       0.77      0.71      0.72       154
weighted avg       0.77      0.76      0.75       154
```

XGBoost

In [32]:
```
pip install -U scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\urvi sharma\anaconda3\lib\site-packages (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\urvi sharma\anaconda3\lib\site-packages (from scikit-learn) (2.
2.0)
Requirement already satisfied: numpy>=1.17.3 in c:\users\urvi sharma\anaconda3\lib\site-packages (from scikit-learn) (1.20.3)
Note: you may need to restart the kernel to use updated packages.Requirement already satisfied: scipy>=1.3.2 in c:\users\urvi s
harma\anaconda3\lib\site-packages (from scikit-learn) (1.7.1)
Requirement already satisfied: joblib>=1.1.1 in c:\users\urvi sharma\anaconda3\lib\site-packages (from scikit-learn) (1.2.0)
```

In [33]:
```
import xgboost
```

In [34]:
```
from xgboost import XGBClassifier
```

In [35]:
```
xg=XGBClassifier()
xg.fit(X_train,y_train)
```

Out[35]:
```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, feature_types=None, gamma=0, gpu_id=-1,
              grow_policy='depthwise', importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_bin=256, max_cat_threshold=64, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0, ...)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [36]:
```
report=classification_report(y_test,xg.predict(X_test))
print('Report:\n',report)
```

```
Report:
               precision    recall  f1-score   support

           0       0.75      0.78      0.76        95
           1       0.62      0.58      0.60        59

    accuracy                           0.70       154
   macro avg       0.68      0.68      0.68       154
weighted avg       0.70      0.70      0.70       154
```

In [38]:
```
# hyper parameter tuning

xg_gs = GridSearchCV(xg, {'n_estimators':range(75, 125),
                          'max_depth':range(1,5)})
xg_gs.fit(X_train, y_train)
```

Out[38]:
```
GridSearchCV(estimator=XGBClassifier(base_score=0.5, booster='gbtree',
                                     callbacks=None, colsample_bylevel=1,
                                     colsample_bynode=1, colsample_bytree=1,
                                     early_stopping_rounds=None,
                                     enable_categorical=False, eval_metric=None,
                                     feature_types=None, gamma=0, gpu_id=-1,
                                     grow_policy='depthwise',
                                     importance_type=None,
                                     interaction_constraints='',
                                     learning_rate=0.300000012, max_bin=256,
                                     max_cat_threshold=64, max_cat_to_onehot=4,
                                     max_delta_step=0, max_depth=6,
                                     max_leaves=0, min_child_weight=1,
                                     missing=nan, monotone_constraints='()',
                                     n_estimators=100, n_jobs=0,
                                     num_parallel_tree=1, predictor='auto',
                                     random_state=0, ...),
             param_grid={'max_depth': range(1, 5),
                         'n_estimators': range(75, 125)})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [39]:
```
xg_gs.best_params_
```

Out[39]:
```
{'max_depth': 1, 'n_estimators': 88}
```

Best XGBoost model

In [41]:
```
from xgboost import XGBClassifier

xg_best = XBGClassifier(max_depth = 1, n_estimators = 88)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
C:\Users\URVISH~1\AppData\Local\Temp/ipykernel_2636/3274697898.py in <module>
      1 from xgboost import XGBClassifier
      2
----> 3 xg_best = XBGClassifier(max_depth = 1, n_estimators = 88)

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

In [ ]:
```
xg_best.fit(X_train, y_train)
```

In [ ]:
```
report = classification_report(y_test, xg_best.predict(X_test))
print('REPORT:\n', report)
```

In [ ]:
```
grad_best.feature_importances_
```

In [ ]:
```
df=pd.DataFrame({'Feature':X.columns, 'Imp':grad_best.feature_importances_})
df
```

In [ ]:
```
df1=df.sort_values(['Imp'],ascending=False)
df1
```

In [ ]:
```
sns.barplot(x=df1['Imp'],y=df1['Feature'],data=df1);
```

In [ ]:
```

```