Boosting

Boosting

Different types:

AdaBoost

GradientBoost

XGBoost

Perform boosting on diabetes dataset

Tune the hyper parameter and get better models

Compare the models.

Adaboost

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

In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:
```
db=pd.read_csv('diabetes.csv')
db
```

Out[3]:

	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 [5]:
```
y=db['Outcome']
X=db.drop(['Outcome'],axis=1)
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X_scaled=scaler.fit_transform(X)
X_scaled
```

Out[5]:
```
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]])
```

In [6]:
```
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)
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[6]:
```
((614, 8), (154, 8), (614,), (154,))
```

Building- AdaBoostCalssifier

In [7]:
```
from sklearn.ensemble import AdaBoostClassifier
abc=AdaBoostClassifier(random_state=10)
abc.fit(X_train,y_train)
```

Out[7]:
```
AdaBoostClassifier(random_state=10)
```

In [8]:
```
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 hyper parameter tuning

In [9]:
```
from sklearn.model_selection import GridSearchCV

abc_gs=GridSearchCV(abc,{'n_estimators':range(25,75),
                         'learning_rate':[0,0.25,0.5,0.75,1]})
```

In [10]:
```
abc_gs.fit(X_train, y_train)
```

```
/opt/anaconda3/lib/python3.9/site-packages/sklearn/model_selection/_validation.py:372: 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 "/opt/anaconda3/lib/python3.9/site-packages/sklearn/model_selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/opt/anaconda3/lib/python3.9/site-packages/sklearn/ensemble/_weight_boosting.py", line 486, in fit
    return super().fit(X, y, sample_weight)
  File "/opt/anaconda3/lib/python3.9/site-packages/sklearn/ensemble/_weight_boosting.py", line 114, in fit
    raise ValueError("learning_rate must be greater than zero")
ValueError: learning_rate must be greater than zero

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/opt/anaconda3/lib/python3.9/site-packages/sklearn/model_selection/_search.py:969: UserWarning: One or more of the test scores
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 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[10]:
```
GridSearchCV(estimator=AdaBoostClassifier(random_state=10),
             param_grid={'learning_rate': [0, 0.25, 0.5, 0.75, 1],
                         'n_estimators': range(25, 75)})
```

In [11]:
```
abc_gs.best_params_
```

Out[11]:
```
{'learning_rate': 0.25, 'n_estimators': 53}
```

In [12]:
```
## 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[12]:
```
AdaBoostClassifier(learning_rate=0.25, n_estimators=53, random_state=10)
```

In [13]:
```
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

In [15]:
```
from sklearn.ensemble import GradientBoostingClassifier
```

In [16]:
```
gbc=GradientBoostingClassifier(random_state=10)
gbc.fit(X_train,y_train)
```

Out[16]:
```
GradientBoostingClassifier(random_state=10)
```

In [17]:
```
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
```

In [18]:
```
# Hyper parameter tuning

gbc_gs=GridSearchCV(gbc,{'n_estimators':range(75,125),
                         'max_depth':range(1,5)})
```

In [20]:
```
gbc_gs.fit(X_train,y_train)
```

Out[20]:
```
GridSearchCV(estimator=GradientBoostingClassifier(random_state=10),
             param_grid={'max_depth': range(1, 5),
                         'n_estimators': range(75, 125)})
```

In [21]:
```
gbc_gs.best_params_
```

Out[21]:
```
{'max_depth': 2, 'n_estimators': 76}
```

Best Gradient Boosting model

In [22]:
```
grad_best=GradientBoostingClassifier(max_depth=2,n_estimators=76,random_state=10)
grad_best.fit(X_train,y_train)
```

Out[22]:
```
GradientBoostingClassifier(max_depth=2, n_estimators=76, random_state=10)
```

In [23]:
```
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 [25]:
```
from xgboost import XGBClassifier
```

```
/opt/anaconda3/lib/python3.9/site-packages/xgboost/compat.py:36: FutureWarning: pandas.Int64Index is deprecated and will be rem
oved from pandas in a future version. Use pandas.Index with the appropriate dtype instead.
  from pandas import MultiIndex, Int64Index
```

In [26]:
```
conda install -c conda-forge xgboost
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

# All requested packages already installed.

Retrieving notices: ...working... done

Note: you may need to restart the kernel to use updated packages.
```

In [27]:
```
xg=XGBClassifier()
xg.fit(X_train,y_train)
```

```
/opt/anaconda3/lib/python3.9/site-packages/xgboost/sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is d
eprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=F
alse when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num
_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[11:25:57] WARNING: /var/folders/sy/f16zz6x50xz3113nwtb9bvq00000gp/T/abs_44tbtwf8c1/croots/recipe/xgboost-split_1659548960882/w
ork/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was
changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

Out[27]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=4,
              num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [28]:
```
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 [29]:
```
# hyper parameter tuning

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

In [30]:
```
xg_gs.fit(X_train,y_train)
```

```
/opt/anaconda3/lib/python3.9/site-packages/xgboost/sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
/opt/anaconda3/lib/python3.9/site-packages/xgboost/sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
/opt/anaconda3/lib/python3.9/site-packages/xgboost/sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
/opt/anaconda3/lib/python3.9/site-packages/xgboost/sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
```

In [31]:
```
xg_gs.best_params_
```

Out[31]:
```
{'max_depth': 1, 'n_estimators': 88}
```

Best XGBoost model

In [32]:
```
xg_best=XGBClassifier(max_depth=1, n_estimators=88)
```

In [33]:
```
xg_best.fit(X_train,y_train)
```

```
[11:32:29] WARNING: /var/folders/sy/f16zz6x50xz3113nwtb9bvq00000gp/T/abs_44tbtwf8c1/croots/recipe/xgboost-split_1659548960882/work/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

Out[33]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=1, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=88, n_jobs=4,
              num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [34]:
```
report=classification_report(y_test,xg_best.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
```

The best model so far is w rt gradient boost

In [35]:
```
grad_best.feature_importances_
```

Out[35]:
```
array([0.0300584 , 0.51152196, 0.01588201, 0.00952595, 0.0293268 ,
       0.1579897 , 0.08449475, 0.16120043])
```

In [36]:
```
df=pd.DataFrame({'Feature':X.columns, 'Imp':grad_best.feature_importances_})
df
```

Out[36]:

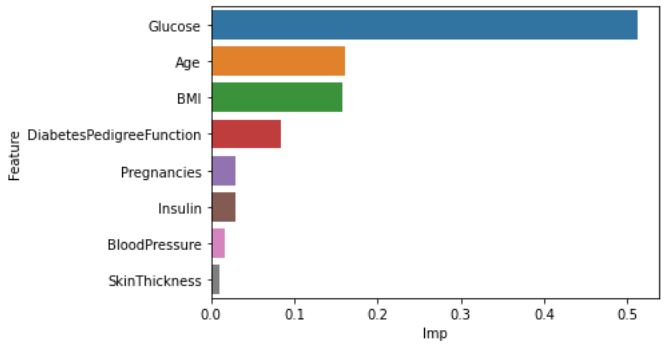
	Feature	Imp
0	Pregnancies	0.030058
1	Glucose	0.511522
2	BloodPressure	0.015882
3	SkinThickness	0.009526
4	Insulin	0.029327
5	BMI	0.157990
6	DiabetesPedigreeFunction	0.084495
7	Age	0.161200

In [38]:
```
df1=df.sort_values(['Imp'],ascending=False)
df1
```

Out[38]:

	Feature	Imp
1	Glucose	0.511522
7	Age	0.161200
5	BMI	0.157990
6	DiabetesPedigreeFunction	0.084495
0	Pregnancies	0.030058
4	Insulin	0.029327
2	BloodPressure	0.015882
3	SkinThickness	0.009526

In [39]:
```
sns.barplot(x=df1['Imp'],y=df1['Feature'],data=df1);
```



In [ ]:
```

```