SURYA GROUP OF INSTITUTION

VIKRAVANDI 605-652

PHASE 2 INNOVATION

PREDICTION HOUSE PRICES USING MACHINE LEARNING

(XG BOOSTER)

NAAN MUDHALVAN

PREPARED BY:

Veenadevi E

REG NO :422221106311

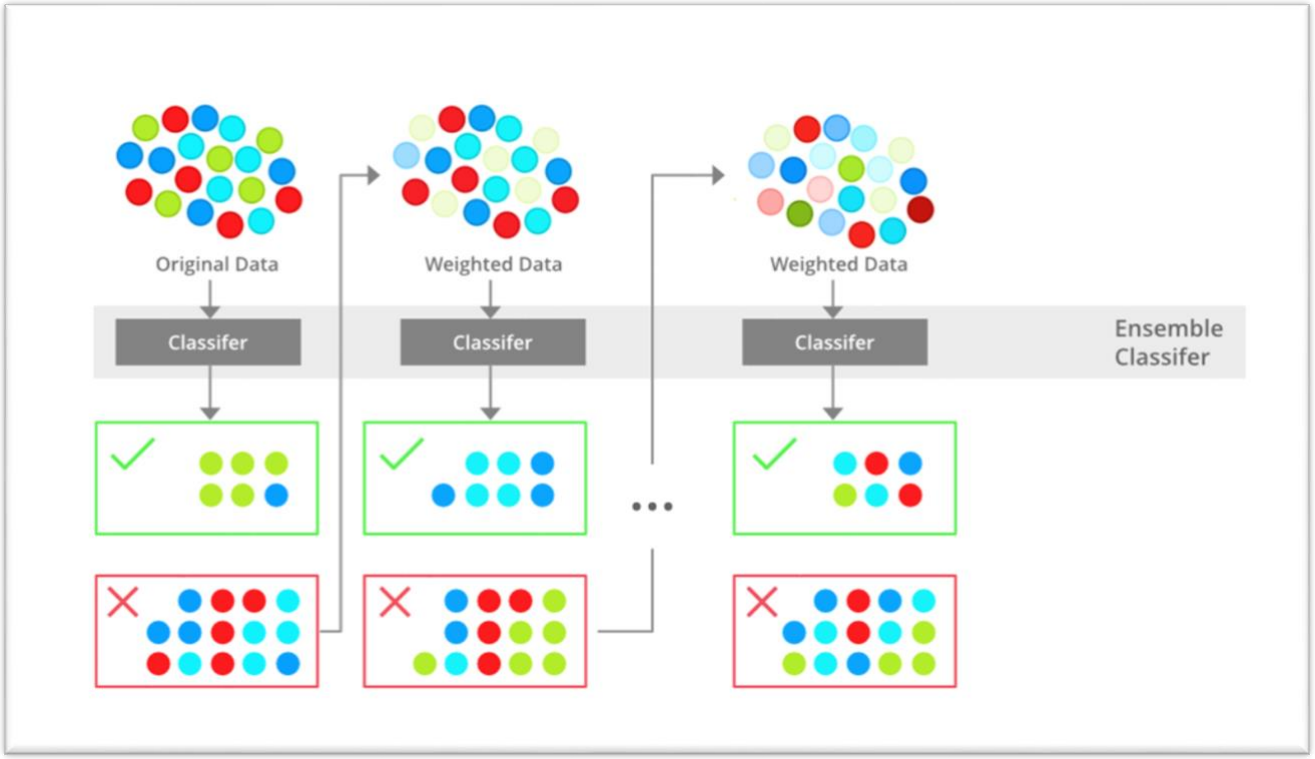
ECE DEPARTMENT

3RD YEAR 5TH SEM

AI_ PHASE 2:

Consider exploring advanced regression techniques like XG Boost for Improved Prediction of House Price Using XGBoost Regression Algorithm 2153 where binary vector is represented as [1,0,0], [1,0,0], [0,1,0], [0,0,1] then fed into model which is easier to understand for most machine learning algorithms. XGBoost regression also uses one hot encoding for understanding categorical values. Fig.1.Flow Diagram for House Price Prediction XGBoost regression is short form for extreme gradient boost regression. It works well compared to others machine learning algorithms.XGBoost is one of the best supervised learning algorithms which can inferred by the way it flows, it consists of objective function and base learners.Loss function is present in objective function which shows the difference between actual values and predicted values whereas regularisation term is used for showing how far is actual value away from predicted value. Ensemble learning used in XGBoost considers many models which are known as base learners for predicting a single value. Not all base learners are expected to have bad prediction so that after summing up all of them bad prediction cancelled out by good prediction. A regressor is the one that fits a model using given features and predicts the unknown output value. Dataset for processing is taken from a Kaggle competition and fed into the model after pre-processing as specified in Figure.

ARCHITECTURE:



ALOGORITHM:

The algorithm for prediction of price of house using XG Boost Regression is specified below.

Types of Boosting Algorithms

In a broad sense, there are three major types of boosting algorithm that finds extensive usage in the ML world.

Adaptive Boosting (Adaboost): It combines the group of weak learner bases to create a strong learner. In the first iteration, it gives equal weight to each data sample. If an incorrect prediction occurs, it gives high weight to that observation. Adaptive Boosting repeats this procedure in the next iteration phase and continues until the desired level of accuracy has been achieved.

Gradient Boosting (GBM): This boosting algorithm works on the principle gradient descent algorithm. Gradient descent is the backbone of modern machine learning procedures which is defined as an iterative optimization process that helps find a local minimum/maximum of a given function. The given function is actually the loss function and Gradient Descent works in an evolving way to minimize the loss function by finding the optimal parameters for the machine learning problem at hand.

Extreme Gradient Boosting (XG Boost): As the name suggests, we can think of this algorithm as the GBM with a booster steroid dosage that works most optimally, super-fast way by prudent use of software as well as hardware combinations. It is a scalable and distributed ML framework that works on the foundation of parallel processing of individual tree models used for classification and regression problems.

MODEL XG BOOST REGRESSOR :

INPUT:

```
from sklearn import datasets
  from sklearn import metrics
  from sklearn.model_selection import train_test_split
  import matplotlib.pyplot as plt
  import seaborn as sns

  plt.style.use("ggplot")

  import xgboost as xgb

dataset = datasets.load_wine()
  X = dataset.data; y = dataset.target
  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)

model = xgb.XGBClassifier()
  model.fit(X_train, y_train)
  print(); print(model)

expected_y  = y_test
```

```
   predicted_y = model.predict(X_test)

print(metrics.classification_report(expected_y, predicted_y))
   print(metrics.confusion_matrix(expected_y, predicted_y))

dataset = datasets.load_boston()
   X = dataset.data; y = dataset.target
   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)

model = xgb.XGBRegressor()
   model.fit(X_train, y_train

expected_y  = y_test
   predicted_y = model.predict(X_test)


print(metrics.r2_score(expected_y, predicted_y))
   print(metrics.mean_squared_log_error(expected_y, predicted_y))

   plt.figure(figsize=(10,10))
   sns.regplot(expected_y, predicted_y, fit_reg=True, scatter_kws={"s": 100})
```

OUTPUT:

```
   XGBClassifier(base_score=0.5, booster="gbtree", colsample_bylevel=1,
     colsample_bynode=1, colsample_bytree=1, gamma=0, learning_rate=0.1,
     max_delta_step=0, max_depth=3, min_child_weight=1, missing=None,
     n_estimators=100, n_jobs=1, nthread=None,
     objective="multi:softprob", random_state=0, reg_alpha=0,
     reg_lambda=1, scale_pos_weight=1, seed=None, silent=None,
     subsample=1, verbosity=1)
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	0.94	0.94	0.94	16
2	0.94	0.94	0.94	18
micro avg	0.96	0.96	0.96	45

macro avg	0.96	0.96	0.96	45
weighted avg	0.96	0.96	0.96	45

```
[[11  0  0]
 [ 0 15  1]
 [ 0  1 17]]
```

0.8359074842658845

0.02822002095090446

PREDICTING PRICES:

```
Prediction5 = model_xg.predict(X_test_scal)
XGBRegressor(base_score=0.5, booster="gbtree", colsample_bylevel=1,
    colsample_bynode=1, colsample_bytree=1, gamma=0,
    importance_type="gain", learning_rate=0.1, max_delta_step=0,
    max_depth=3, min_child_weight=1, missing=None, n_estimators=100,
    n_jobs=1, nthread=None, objective="reg:linear", random_state=0,
    reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
    silent=None, subsample=1, verbosity=1)
```

CONCLUSION:

This project entitled “House Price Prediction Using XG Boost Regression Model.” is useful in buying the houses, by predicting house prices, and thereby to guide their buyers accor dingly. The proposed system is also useful to the buyers to predict the cost of house according to the area it is present. XG boosting algorithm has high accuracy value when compared to all other algorithms regarding house price prediction .