1)What is Gradient Boosting Regression?

Ans- Gradient boosting Regression calculates the difference between the current prediction and the known correct target value. This difference is called residual. After that Gradient boosting Regression trains a weak model that maps features to that residual.

2) Q2. Implement a simple gradient boosting algorithm from scratch using Python and NumPy. Use a simple regression problem as an example and train the model on a small dataset. Evaluate the model's performance using metrics such as mean squared error and R-squared.

Ans- # explore gradient boosting number of trees effect on performance

from numpy import mean

from numpy import std

from sklearn.datasets import make\_classification

from sklearn.model\_selection import cross\_val\_score

from sklearn.model\_selection import RepeatedStratifiedKFold

from sklearn.ensemble import GradientBoostingClassifier

from matplotlib import pyplot

# get the dataset

def get\_dataset():

X, y = make\_classification(n\_samples=1000, n\_features=20, n\_informative=15, n\_redundant=5, random\_state=7)

return X, y

# get a list of models to evaluate

def get\_models():

models = dict()

# define number of trees to consider

n\_trees = [10, 50, 100, 500, 1000, 5000]

for n in n\_trees:

models[str(n)] = GradientBoostingClassifier(n\_estimators=n)

return models

# evaluate a given model using cross-validation

def evaluate\_model(model, X, y):

# define the evaluation procedure

cv = RepeatedStratifiedKFold(n\_splits=10, n\_repeats=3, random\_state=1)

# evaluate the model and collect the results

scores = cross\_val\_score(model, X, y, scoring='accuracy', cv=cv, n\_jobs=-1)

return scores

# define dataset

X, y = get\_dataset()

# get the models to evaluate

models = get\_models()

# evaluate the models and store results

results, names = list(), list()

for name, model in models.items():

# evaluate the model

scores = evaluate\_model(model, X, y)

# store the results

results.append(scores)

names.append(name)

# summarize the performance along the way

print('>%s %.3f (%.3f)' % (name, mean(scores), std(scores)))

# plot model performance for comparison

pyplot.boxplot(results, labels=names, showmeans=True)

pyplot.show()

3) Experiment with different hyperparameters such as learning rate, number of trees, and tree depth to optimise the performance of the model. Use grid search or random search to find the best hyperparameters

Ans-

# random search logistic regression model on the sonar dataset

from scipy.stats import loguniform

from pandas import read\_csv

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import RepeatedStratifiedKFold

from sklearn.model\_selection import RandomizedSearchCV

# load dataset

url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/sonar.csv'

dataframe = read\_csv(url, header=None)

# split into input and output elements

data = dataframe.values

X, y = data[:, :-1], data[:, -1]

# define model

model = LogisticRegression()

# define evaluation

cv = RepeatedStratifiedKFold(n\_splits=10, n\_repeats=3, random\_state=1)

# define search space

space = dict()

space['solver'] = ['newton-cg', 'lbfgs', 'liblinear']

space['penalty'] = ['none', 'l1', 'l2', 'elasticnet']

space['C'] = loguniform(1e-5, 100)

# define search

search = RandomizedSearchCV(model, space, n\_iter=500, scoring='accuracy', n\_jobs=-1, cv=cv, random\_state=1)

# execute search

result = search.fit(X, y)

# summarize result

print('Best Score: %s' % result.best\_score\_)

print('Best Hyperparameters: %s' % result.best\_params\_)

4) What is a weak learner in Gradient Boosting?

Ans- Weak learners are models that perform slightly better than random guessing. Strong learners are models that have arbitrarily good accuracy. Weak and strong learners are tools from computational learning theory and provide the basis for the development of the boosting class of ensemble methods.

5) What is the intuition behind the Gradient Boosting algorithm?

Ans- In gradient boosting, we predict and adjust our predictions in the opposite (negative gradient) direction. This achieves the opposite (minimize the loss). Since, the loss of a model inversely relates to its performance and accuracy, doing so improves its performance.

6) How does Gradient Boosting algorithm build an ensemble of weak learners?

Ans- This algorithm starts by building a decision stump and then assigning equal weights to all the data points. Then it increases the weights for all the points which are misclassified and lowers the weight for those that are easy to classify or are correctly classified.

7) What are the steps involved in constructing the mathematical intuition of Gradient Boosting algorithm?

Ans-

Step 1: Calculate the average of the target label. ...

Step 2: Calculate the residuals. ...

Step 3: Construct a decision tree. ...

Step 4: Predict the target label using all of the trees within the ensemble. ...

Step 5: Compute the new residuals.