Experiment-5: Develop a program for Bias, Variance, Remove duplicates , Cross Validation

**#bias, variance**

import numpy as np # linear algebra

from mlxtend.evaluate import bias\_variance\_decomp

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.neighbors import KNeighborsRegressor

from sklearn.metrics import confusion\_matrix

from sklearn import metrics

from sklearn.linear\_model import LinearRegression

import matplotlib.pyplot as plt

%matplotlib inline

data=pd.read\_csv('/content/sample\_data/heart.csv')

# separate into inputs and outputs

data = data.values

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

# split the data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=1)

# define the model

model = LinearRegression()

# estimate bias and variance

mse, bias, var = bias\_variance\_decomp(model, X\_train, y\_train, X\_test, y\_test, loss='mse', num\_rounds=200, random\_seed=1)

# summarize results

print('MSE: %.3f' % mse)

print('Bias: %.3f' % bias)

print('Variance: %.3f' % var)

#Output



**#Removing duplicates**

The datasets are first merged into a single dataframe, 'iris\_df', using the concat() method. Then, the duplicates are removed from the merged dataset using the drop\_duplicates() method.

**#Removing duplicates**

import pandas as pd

from sklearn.datasets import load\_iris

iris\_data = load\_iris()

iris\_df = pd.DataFrame(iris\_data.data, columns=iris\_data.feature\_names)

print('Shape\n',iris\_df.shape)

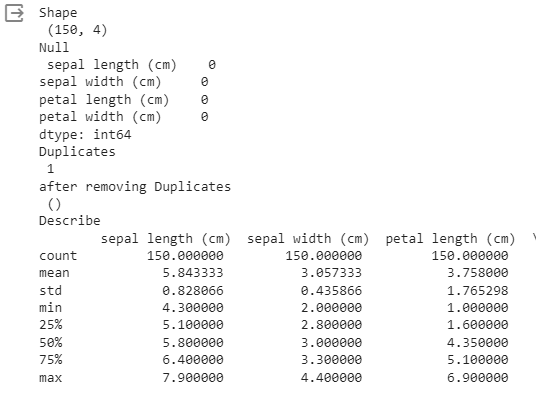
print('Null\n',iris\_df.isnull().sum())

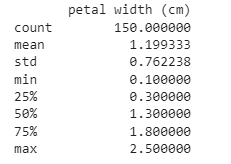
print('Duplicates\n',iris\_df.duplicated().sum())

print('Duplicates\n',iris\_df.duplicated().sum())

print('after removing Duplicates\n',iris\_df.duplicated().sum().shape)print('Describe\n',iris\_df.describe())

Output:





**#k-fold Cross Validation**

from sklearn import datasets

from sklearn.tree import DecisionTreeClassifier

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

X, y = datasets.load\_iris(return\_X\_y=True)

clf = DecisionTreeClassifier(random\_state=42)

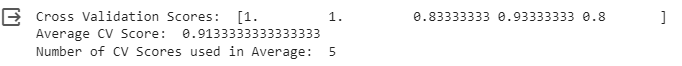
k\_folds = KFold(n\_splits = 5)

scores = cross\_val\_score(clf, X, y, cv = k\_folds)

print("Cross Validation Scores: ", scores)

print("Average CV Score: ", scores.mean())

print("Number of CV Scores used in Average: ", len(scores))



**#stratified cross fold**

from sklearn import datasets

from sklearn.tree import DecisionTreeClassifier

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

X, y = datasets.load\_iris(return\_X\_y=True)

clf = DecisionTreeClassifier(random\_state=42)

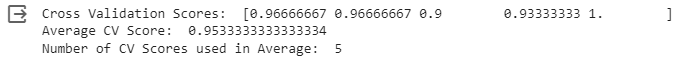
sk\_folds = StratifiedKFold(n\_splits = 5)

scores = cross\_val\_score(clf, X, y, cv = sk\_folds)

print("Cross Validation Scores: ", scores)

print("Average CV Score: ", scores.mean())

print("Number of CV Scores used in Average: ", len(scores))



**#Leave-one-out-cut**

from sklearn import datasets

from sklearn.tree import DecisionTreeClassifier

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

X, y = datasets.load\_iris(return\_X\_y=True)

clf = DecisionTreeClassifier(random\_state=42)

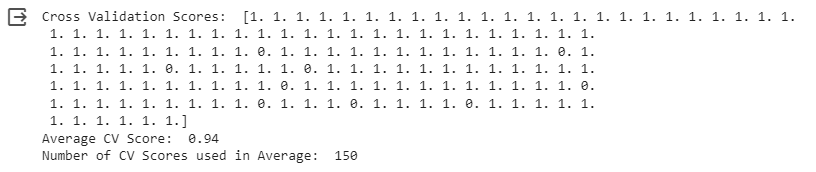
loo = LeaveOneOut()

scores = cross\_val\_score(clf, X, y, cv = loo)

print("Cross Validation Scores: ", scores)

print("Average CV Score: ", scores.mean())

print("Number of CV Scores used in Average: ", len(scores))



#Leave-P-out cross validation

from sklearn import datasets

from sklearn.tree import DecisionTreeClassifier

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

X, y = datasets.load\_iris(return\_X\_y=True)

clf = DecisionTreeClassifier(random\_state=42)

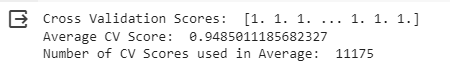
lpo = LeavePOut(p=2)

scores = cross\_val\_score(clf, X, y, cv = lpo)

print("Cross Validation Scores: ", scores)

print("Average CV Score: ", scores.mean())

print("Number of CV Scores used in Average: ", len(scores))



**#Shuffle\_split cross validation**

from sklearn import datasets

from sklearn.tree import DecisionTreeClassifier

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

X, y = datasets.load\_iris(return\_X\_y=True)

clf = DecisionTreeClassifier(random\_state=42)

ss = ShuffleSplit(train\_size=0.6, test\_size=0.3, n\_splits = 5)

scores = cross\_val\_score(clf, X, y, cv = ss)

print("Cross Validation Scores: ", scores)

print("Average CV Score: ", scores.mean())

print("Number of CV Scores used in Average: ", len(scores))

