**ARTIFICIAL INTELLIGENCE (18CSC305J) LAB**

EXPERIMENT 12

Development of ensemble model for an application

**Harsh Goel**

**RA1811003010185**

**CSE-C1**

**Aim:**

To Develop a model an ensemble model for an application.

**Language:** Python

**Theory:**

Ensemble learning helps improve machine learning results bycombining several models. This approach allows the production of betterpredictive performance compared to a single model. Basic idea is to learna set of classifiers (experts) and to allow them to vote.

**Problem Formulation and Algorithm:**

* Import the required libraries.
* Import the data file in the program.
* Clean the data.
* Find the required features on which the model predicts.
* Split the data into two parts say train\_x and train\_y
* Train the model using the data train\_x and find the mean absolute error with the predicted value and train\_y.
* We use random forest and decision trees algorithms to predict our model.
* A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as bagging.

**Source Code:**

import pandas as pd

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error

from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import mean\_absolute\_error

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

melbourne\_file\_path = './melb\_data.csv'

melbourne\_data = pd.read\_csv(melbourne\_file\_path)

melbourne\_data.columns

melbourne\_data = melbourne\_data.dropna(axis=0)

y = melbourne\_data.Price

melbourne\_features = ['Rooms', 'Bathroom', 'Landsize', 'Lattitude',

'Longtitude']

X = melbourne\_data[melbourne\_features]

X.describe()

X.head()

# Define model. Specify a number for random\_state to ensure same results each run

melbourne\_model = DecisionTreeRegressor(random\_state=1)

# Fit model

melbourne\_model.fit(X, y)

print("Making predictions for the following 5 houses:")

print(X.head())

print("The predictions are")

print(melbourne\_model.predict(X.head()))

predicted\_home\_prices = melbourne\_model.predict(X)

mean\_absolute\_error(y, predicted\_home\_prices)

# split data into training and validation data, for both features and target

# # The split is based on a random number generator. Supplying a numeric value to

# the random\_state argument guarantees we get the same split every time we

# run this script.

train\_X, val\_X, train\_y, val\_y = train\_test\_split(X, y, random\_state =0)

# Define model

melbourne\_model = DecisionTreeRegressor()

# Fit model

melbourne\_model.fit(train\_X, train\_y)

# get predicted prices on validation data

val\_predictions = melbourne\_model.predict(val\_X)

print(mean\_absolute\_error(val\_y, val\_predictions))

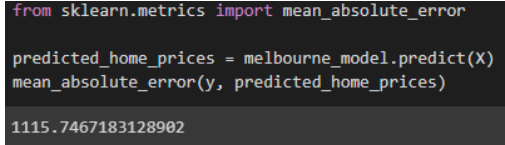
forest\_model = RandomForestRegressor(random\_state=1)

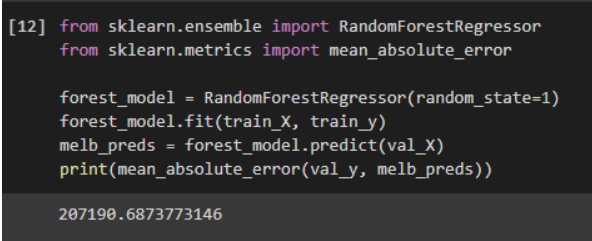
forest\_model.fit(train\_X, train\_y)

melb\_preds = forest\_model.predict(val\_X)

print(mean\_absolute\_error(val\_y, melb\_preds))

**Output Screenshots:**





**Verification:**

From both the algorithms we can see the difference in the mean absolute error. The predictions made are nearby the data which are input, so we can verify the data is correctly predicted as it lies along the dataset provided.

**Result:** Hence, successfully implemented the problem and verified the output and document result.