ML prediction

Machine prediction is the process of using historical data to train a numerical model which can then make prediction or forecast about new unseen data.

```
#Importing required libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score,mean_squared_error
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import ExtraTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBRegressor
import pickle
```

- Pandas for data manipulation.
- Train_test_split it splits the datasets into training and testing sets for model evaluation.
- R2, mse & mae these are Evaluation metrics used for regression model performance
- RandomForestRegressor algorithm is a ensemble method build multiple dt combines there prediction for accuracy and reduce overfitting.
- DecisionTreeRegressor algorthim recursively split data till left the leaf node and it feature make decision.
- Gradient boosting used to reduce the errors and learning grade slowly.
- XGBRegressor optimized and efficient implementation of gradient boosting regression algorithm provided by the XGBoost library.
- pickle: This module is used for serializing and deserializing Python objects and for saving and loading machine learning models.

```
#reading preprocessed dataset
df1 = pd.read_csv('/content/drive/MyDrive/Singapore Resale /final.csv')
```

- Reading the final csv from the specific path and loads it's a content into pandas Data Frame of df1.

```
df1
```

- This code displays the Data Frame ‘df1’ allowing for a quick inspection of the data's structure and content.

Training the models

```
x = df1[['selling_month','selling_year','town_code','storey_min','storey_max','floor_area_sqm','flat_modelcode','lease_commence_date']]
y = df1['resale_price']
```

- splitting the dataset into x features variable extracted from the data frame df1.
- Y is a target variable where the x is independent and y is dependent variable for building the predictive model.

```
#displaying the shape
x.shape,y.shape
```

- It displays the x and y shapes for understanding the size of the datasets.
- To verify the size of the structure and helps to see mismatched or missing data which could affect the model performance.

Selecting the model

```
#using function method we can check the best models
def model_regression(xtrain,xtest,ytrain,ytest,algorithm):

  for i in algorithm:
    model = i().fit(xtrain,ytrain)

    #predict for accuracy
    y_train_pred = model.predict(xtrain)

    y_test_pred = model.predict(xtest)

    #r2score prection
    r2_train = r2_score(ytrain,y_train_pred)

    r2_test = r2_score(ytest,y_test_pred)

    data = {'Algorthim':i.__name__, 'Training_r2_score':r2_train, 'Testing_r2_score':r2_test}

    print(data)
```

- Defining the function named as model_regression and iterates through a list of regression algorithms specified by algorithm.
- For each algorithm in the list trains a model on training data ‘xtrain’,'ytrain’.
- It predicts the target variables for both training and testing datasets.
- Calculates the R2 score for both training and testing predictions of ‘r2_train’ and ‘r2_test’.
- It displays using print function algorithm names along with training and testing r2 score.
- It helps to find best algorithm based on the performance metrics.

```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2)
```

- The feature variable ‘x’ and target variable ‘y’ into training and testing sets using the train_test_split.
- It allocates 80% of training(x_train,y_train)data and 20% for testing(x_test,y_test)data.
- Using this partition datasets into separate subset train and test model it helps to evaluating the performance of the model unseen data.

```
model_regression(xtrain,xtest,ytrain,ytest,[DecisionTreeRegressor,ExtraTreeRegressor,RandomForestRegressor,GradientBoostingRegressor,XGBRegressor])
```

```
{'Algorthim': 'DecisionTreeRegressor', 'Training_r2_score': 0.998304223221716 2, 'Testing_r2_score': 0.9620463379343377}
{'Algorthim': 'ExtraTreeRegressor', 'Training_r2_score': 0.9983016073997889, 'Testing_r2_score': 0.945545933655 1869}
{'Algorthim': 'RandomForestRegressor', 'Training_r2_score': 0.99568867997583 57, 'Testing_r2_score': 0.9776806282263405}
{'Algorthim': 'GradientBoostingRegressor', 'Training_r2_score': 0.9077955401239038, 'Testing_r2_score': 0.9088034414451206}
{'Algorthim': 'XGBRegressor', 'Training_r2_score': 0.9740723063731437, 'Testing_r2_score': 0.9736988234355136}
```

- Calling the function of model_regression and the following arguments.
- The function trains each algorithm on the training data, predicts the target variable for both training and testing datasets.
- DecisionTreeRegressor algorthim recursively split data till left the leaf node and it feature make decision.
- ExtraTreeRegressor by building multiple dt with randomization at each step extra tree reduce the variance and overfitting occur with individual decision tree.

- RandomForestRegressor algorithm is a ensemble method build multiple dt combines there prediction for accuracy and reduce overfitting.
- Gradient boosting used to reduce the errors and learning grade slowly.
- XGBRegressor optimized and efficient implementation of gradient boosting regression algorithm provided by the XGBoost library
- The r2_score for each algorithm's predictions, and prints the results.
- We can compare the algorithms and choose the best fit.
- Here we choose random forest for best score.

```
model = RandomForestRegressor(n_estimators = 10)

model.fit(xtrain,ytrain)
y_pre_train = model.predict(xtrain)
y_pre_test = model.predict(xtest)

#r2 score prediction
r2_train = r2_score(ytrain, y_pre_train)
r2_test = r2_score(ytest, y_pre_test)

print(f'Training_r2_score: {r2_train}, Testing_r2_score: {r2_test}')
```

```
Training_r2_score: 0.9944985854675606, Testing_r2_score: 0.9756693436651825
```

- The algorithm assigned to model with 10 estimators decision tree and the model is trained on the training data ‘x_train’ and ‘y_train’ using the fit method.
- Prediction are made both training and testing datasets using the predict method.
- Calculates the R2 score for both training and testing predictions of ‘r2_train’ and ‘r2_test’.
- It displays the r2 score using print function.

```
model.fit(xtrain,ytrain)
```

- Loading the model xtrain and ytrain Once trained, the model can be used to make predictions on new, unseen data.

Checking the Trained models

```
x.loc[6].tolist()
```

```
[1.0, 1990.0, 0.0, 7.0, 9.0, 67.0, 12.0, 1977.0]
```

```
y.loc[6].tolist()
```

```
42000.0
```

- It retrieves the data of the 6th location in the DataFrame x and y converts it into a list format.

```
predicted = model.predict([x.loc[6].tolist()])
predicted[0]
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/b
  warnings.warn(
39673.33333333333
```

- It predicts the resale price for given value it will predict using the model.
- we use [0] to get the first and only element of the array which represent predicted value.

```
predicted = model.predict([x.loc[4].tolist()])
predicted[0]
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/
  warnings.warn(
44600.0
```

- predicts the resale price for a specific data of 4th data location displaying predicted value using the model.

Saving the model

```
import pickle
with open('S1_regression.pkl', 'wb') as f:
    pickle.dump(model, f)
```

- Saving the model using dump function of S1_regression.pkl file through pickle module.
- By saving the binary format we can later load reuse for making prediction don't need to run from scratch.

```
with open("/content/drive/MyDrive/Singapore Resale /S1_regression.pkl", 'rb') as f:
  model = pickle.load(f)
```

```
model
```

```
RandomForestRegressor
RandomForestRegressor(n_estimators=10)
```

- Reading the binary file where pickle loaded in specific file machine learning model stored in the file, assigning it to the variable model.