Homework 1 submission

ECET 612 — Applied Machine Learning

Murat Isik

April 16, 2022

1 Submitted files

For this assignment, besides this report, the following archives were created:

1.1 SRC Folder

+ ***”utils.py”***: This **script** contains all the necessary functions to generate the clusters, animations, plots, etc. It contains the following functions.

    * *” load_housing_data”*: This **function** gets input from the dataset.
    * *” matplotlib.pyplot”*: This **function** draws a graph based on variables.
    * *” scatter matrix.plotting”*: This **function** is the scatter plot matrix plotting all the pairwise scatter between different variables in the form of a matrix.
    * *” train_test_split”*: This **function** split arrays or matrices into random train and test subsets
    * *” Simple Imputer”*: This **function** imputation transformer for completing missing values.
    * *” Ordinal Encoder”*: This **function** encodes categorical features as an integer array.
    * *” OneHot Encoder”*: This **function** encodes categorical features as a one-hot numeric array.
    * *” Base Estimator”*: This **function** base class for all estimators in scikit-learn.
    * *” Pipeline”*: This **function** transforms with a final estimator.
    * *” Standard Scaler”*: This **function** standardizes features by removing the mean and scaling to unit variance.
    * *” Column Transformer”*: This **function** applies transformers to columns of an array or pandas Data Frame.
    * “Linear Regression”: This **function** ordinary least squares Linear Regression
    * “Mean_Squared_Error”: This **function** estimator measures the average of error squares.
    * “Decision Tree Regressor”: This **function** is used to fit a sine curve with additional noisy observation.
    *
    * ”+ ***”chapter2.ipyb”***: This **script** initiates all the necessary function calls, and also generates the graphs for housing prices. First, it calculates all the housing prices with the information provided by the CSV file. Then it finds the plot of the graphs by using the *different python* functions. Afterward, it creates housing predictions of the price.

2 Discussion

The confirmation set is used to estimate a particular model, however, it is only utilized on a regular basis. As machine literacy experts, we use this information to fine-tune the model hyperparameters. As a result, the model sees this data from time to time but never "learns" from it. We utilize the findings of the confirmation set to update the advanced position hyperparameters. As a result, the confirmation set has an effect on a model, but only laterally. The confirmation set is often referred to as the Dev or Development set. This makes sense because this dataset is useful during the model's "development" stage. In Figure 1, the validation set workflow is explained with the trained model and validation model. This tweak model is according to the validation set results.

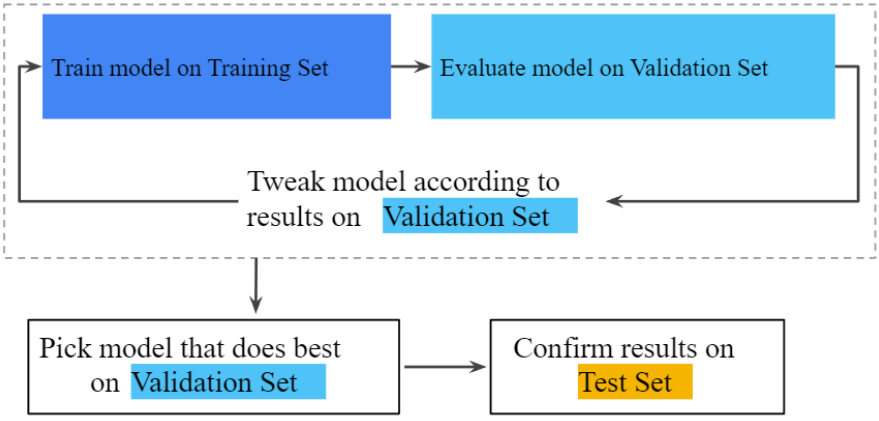

Figure 1 Validation set to evaluate results from the training set [1]

3 Results

3.1 Include training and validation accuracies.

We need to get results one by one with different ML algorithms covered in the class.

Linear Regression;

Straight relapse solves the problem of predicting subordinate variable esteem (y) based on a given free variable (x). As a result, this relapse technique discovers a direct association between x (input) and y. (output). As a result, the title is Straight Relapse. If we plot the free variable (x) on the x-axis and the subordinate variable (y) on the y-axis, straight relapse offers us a straight line that best fits the data points, as seen in the picture below.

```
dataset = pd.read_csv('stock.csv')
dataset.shape
dataset.describe()
dataset.plot(x='High', y='Low', style='o')
plt.title('High vs Low')
plt.xlabel('Low')
plt.ylabel('High')
plt.show()
plt.figure(figsize=(15,10))
plt.tight_layout()
seabornInstance.distplot(dataset['High'])
X = dataset['Low'].values.reshape(-1,1)
y = dataset['High'].values.reshape(-1,1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
regressor = LinearRegression()
regressor.fit(X_train, y_train) #training the algorithm

#To retrieve the intercept:
print(regressor.intercept_)

#For retrieving the slope:
print(regressor.coef_)

y_pred = regressor.predict(X_test)
```

Figure 2 Linear Regression Code

Figure 3 Linear Regression

Decision Tree;

Choice tree relapse observes the highlights of a question and trains a demonstration inside the structure of a tree to predict information in the future to produce significant continuous output. Persistent yield suggests that the output/result is not discrete, that it is not represented just by a discrete, well-known collection of numbers or values. We shall indicate in the graphic our approve and forecast dataset.

```
[12] # Compiling a decision tree model with max_depth = 5 and fitting the model

     dt = DecisionTreeRegressor(max_depth=5)
     dt.fit(X_train, y_train)

     DecisionTreeRegressor(max_depth=5)

     # Making prediction on the validation data and calculating scores (rmse and r2)

     predicts = dt.predict(X_valid)

     print('Validation RMSE = ', np.sqrt(mean_squared_error(y_valid,predicts)))
     print('validation R2 Score = ', r2_score(y_valid, predicts))

     Validation RMSE =  0.023800649138078493
     validation R2 Score =  0.9069808456871286
```

Figure 4 Decision Tree Code

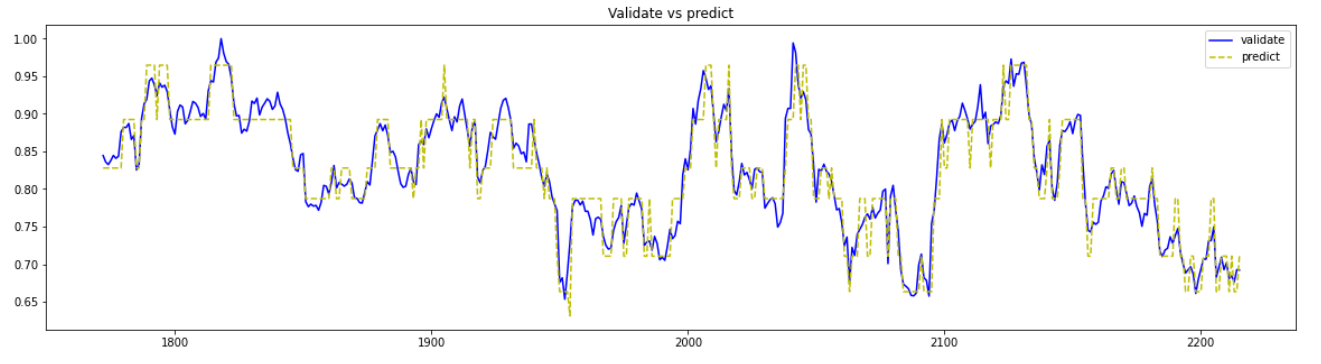

Figure 5 Results of Decision Tree

3.2 State clearly your estimate of the algorithm's performance in terms of RMSE and provide the basis for the performance prediction of the model

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how to spread out these residuals. In other words, it tells you how concentrated the data is around the line of best fit. Root mean square error is commonly used in climatology, forecasting, and regression analysis to verify experimental results.

Root Mean Square Error (RMSE) is a standard way to measure the error of a model in predicting quantitative data. Formally it is defined as follows [2]:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

The output value you get is in the same unit as the required output variable which makes interpretation of loss easy. It is not that robust to outliers as compared to MAE for performing RMSE we have to NumPy square root function over MSE. Most of the time people use RMSE as an evaluation metric and mostly when you are working with deep learning techniques the most preferred metric is RMSE.