Technical Documentation

Project: Shipping Cost Prediction using Machine Learning

Editor used: Visual Studio Code

Hardware Specifications: I5 processor, Windows 10, 16gb Ram

Software specifications: Python 3.8

Folder: Schneker ML Task

Folders contain multiple files:

1. new_task.py
2. demo_new_task.py
3. cost_predictions.csv
4. randomforest.pkl
5. requirements.txt

The **file main.py** contains the model pipeline with multiple helper functions (main task code)

The **file predictions.py** contains the instance to call the Model pipeline along with train and test data file paths. User has choice to enter the models they want to try. Either two models (Random Forest Regressor & XGboost Regressor) can be used for training, or both can be used.

The file **cost_predictions.csv** contains the model predictions obtained for test_data.csv

File **randomforest.pkl** is the file which contains saved model parameters which can be used for further testing.

requirements.txt file has all the packages needed to run the main code

Dataset information:

train_data.csv (has 9 columns, 251156 rows)

test_data.csv (has 8 columns, 62790 rows)

dependent feature: “cost”

independent features: origin_latitude; origin_longitude; destination_latitude; destination_longitude; weight; loading_meters; is_adr; shipping_date

Model Pipeline Explanation

Step 1: Import the necessary packages to run the models and graphs

Step 2: Building the model pipeline called "Shipment_pipeline" using Python Object oriented Classes

Detailed explanations for each function given in this document.

load_data:

This function loads the data files (CSV Files) given by user with Pandas Library as Data frames. As the dataframe contain Shipping_date column, it is converted to date object using Parse Dates function

preprocess_data:

In this function, the necessary preprocessing operations applied on the datasets given by user i.e Train data csv file and Test data csv file. First, the shipping_date column further split to Month, weekday and year columns using Pandas Library.

Using the "origin_latitude", "origin_longitude", "destination_latitude" and "destination_longitude", distance between origin and destination is calculated using Haversine Formula.

Python has package available called haversine to calculate distance between coordinates. This step will be help to improve model performance because, regression models can't handle coordinates directly as features, and most importantly cost is directly corelated to distance. More the distance, more the cost.

Using the haversine package and pandas, distance calculated and added as new feature for the datasets

Then shipping_date, origin_latitude, origin_longitude, destination_latitude, destination_longitude columns are dropped from dataframe.

Distance values are obtained in unit Kilometers, then those values are normalized between 0 and 1

Then further checked data information like shape of datafiles, null values, and data statistics.

eda_data:

In this function, I have performed Explanatory Data Analysis using various graphs.

First scatter plots are plotted comparing “cost” with rest of the columns to get insights, how independent features are related with dependent features

Next, density distribution plots define, how data is distributed among all the columns of dataset, how the data values of columns ranged to get better insights of data.

Box plots are used to identify the outliers in data. Using box plots, all the features are plotted to check whether any outliers are present in data given.

Next, Inter Quartile Range (IQR method) used to find percentage of outliers among all the columns of dataset. As the dataset has very minimal outliers, so I did not proceed to remove them, instead used Random Forest model, which can efficiently perform efficiently against outliers.

Correlation graph to understand how different data columns are related to each other, whether they are positively co-related or negatively corelated to each other. From the correlation graph, except cost and distance, all the other columns have very little correlation, there is no influence of one on other, some of the feathers are negatively correlated.

data_split:

After data preprocessing and explanatory data analysis, I have divided the training dataset into training and test features (X,y).

Further these X,y features are divided into “X_train “, “y_train”, “X_test” , “y_test” with ratio of 65:35

model_selection :

This function explains the model selection part. At the beginning, user has free hands to give models where user want to try. In this task, two models Random Forest and XGBoost Regressors. I have used two models for training the task. Particularly, in this function, if the user gives one model, the pipeline selects only one model for training, if user opts for two models, pipeline considers two models for training.

For the selected model, instance for the model created with necessary hyper parameters.

fit_data:

for the selected model, train data (X_train, y_train) is fit for training using model.fit; then resulted trained model stored using variable.

predict_data:

Once the model has trained with training data, the trained model is called in to this function and model.predict is applied on X_test data. The resulted model predictions are stored using dictonary as key value pairs with key as model name and values as predicted scores.

evaluate_data:

for any model evaluation, we should verify how accurate model predictions are, compared to ground truth data available. In this function, different I have chosen three metrics to compare predicted data with y_test data.

Mean absolute error,

Root Mean Square

R square error (important metric)

All these metrics are applied with predicted values and y_test values, we get scores for each metric for 2 models' random forest and xgboost.

The obtained results are stored as dataframe with metric names as index column.

visualize:

This function takes the dataframe from above function which has validation scores, then plotted the results using bar chart.

save_model :

To evaluate the task, r square metric will be used. So, from the results dataframe, which obtained from evaluate_data function, the model which got the highest r square score is considered and saved as pickle file.

eval_test_data:

This final function, takes the test data frame, applies the preprocessing steps for the test data frame. Then I have loaded the saved pickle file, and applied pre processed test dataframe to the saved model pickle file using model.predict command.

Resulting predictions are stored in "cost_predictions.csv" file.