Configuration Manual

MSc Research Project
Data Analytics

Sasikumar Jayapal
Student ID: x21153272

School of Computing
National College of Ireland

Supervisor: Cristina Hava Muntean

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Sasikumar Jayapal
Student ID:	x21153272
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Cristina Hava Muntean
Submission Due Date:	15/12/2022
Project Title:	Configuration Manual
Word Count:	1065
Page Count:	11

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Sasikumar Jayapal
Date:	29th January 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sasikumar Jayapal
x21153272

1 Introduction

This document includes comprehensive instructions for setting up the hardware and software setups, as well as detailed instructions for carrying out the research work including, dataset preparation, preprocessing, model building, and evaluation.

2 Hardware and Software Requirements

2.1 Hardware Configuration

This research work has been carried out on a personal laptop, hence the following figure 1 depicts the system configuration setup. The hardware configuration setup is Intel Core i7 processor, 8GB of RAM, and a 64-bit operating system.

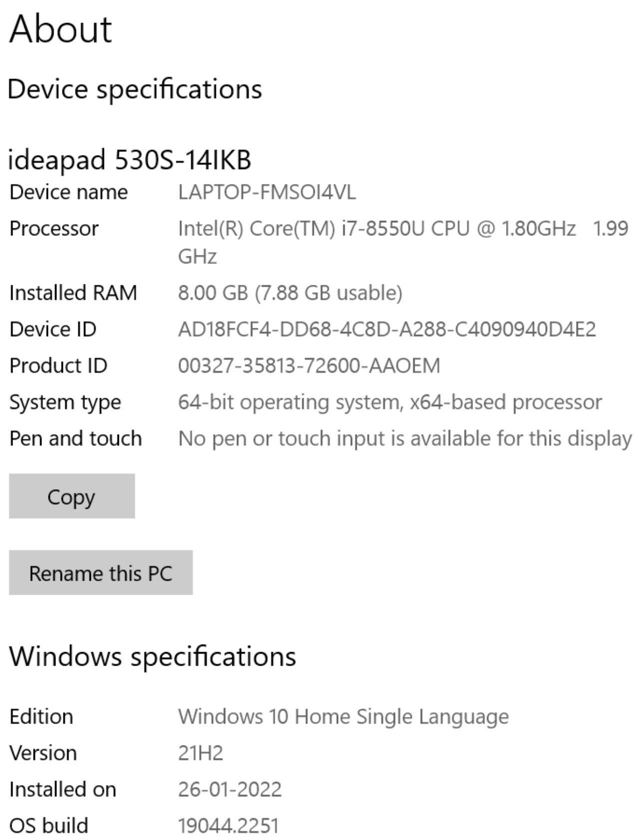

Figure 1: System Configuration

2.2 Software Configuration

This section describes the environments that were set up and used for the implementation; these should be prepared in advance. the following software or applications are configured and they should be configured on the system beforehand.

1. Jupyter Notebook 6.4.5
2. Google Colaboratory (Cloud-based Jupyter notebook environment)
3. Python 3.9.7
4. Microsoft Office 2018: Word, Excel, PowerPoint
5. Online LaTex editor overleaf
6. Google Chrome and Microsoft Edge

3 Methodology and Implementation

3.1 Dataset Collection and Preparation

- Step1: The dataset for the research work has been collected from the public repository called kaggle as shown in the below figure 2.

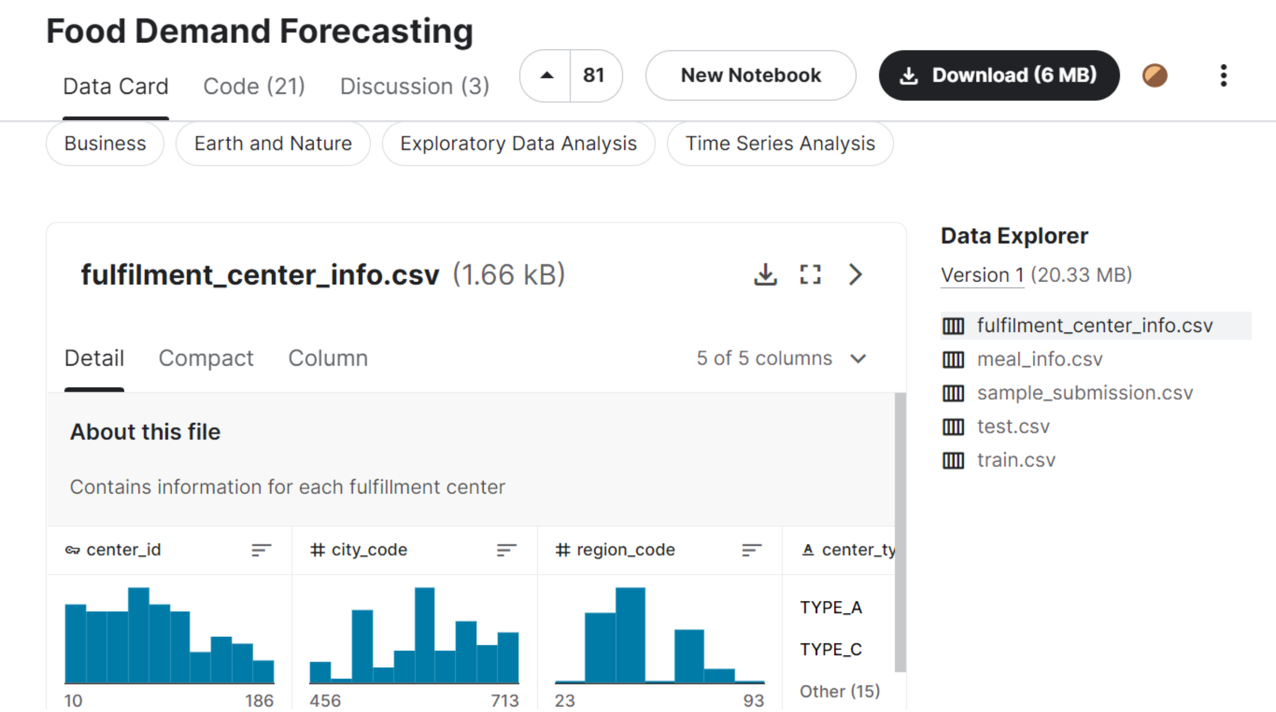

Figure 2: Dataset Collection

- Step2: The dataset contains three files meal_info.csv, fulfilment_center_info.csv, and Train.csv, and all three of them were collected and stored at the local drive for the implementation as shown in the Figure 3.

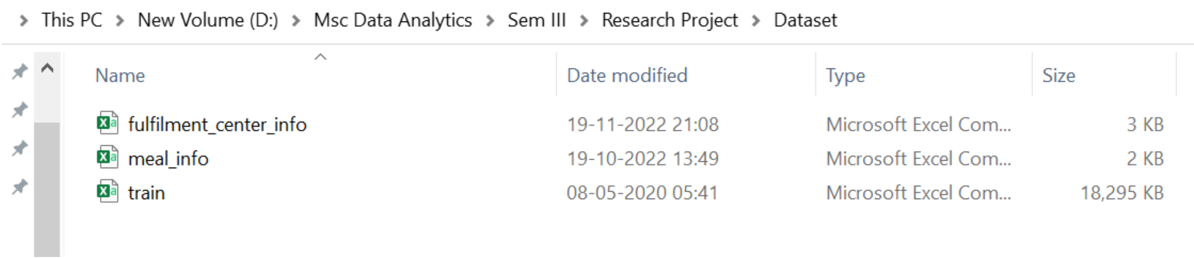

Figure 3: Dataset

- Step3: Another copy of the dataset files has been uploaded to google drive and configured to be accessed from the Google Collaboratory as shown in the Figure

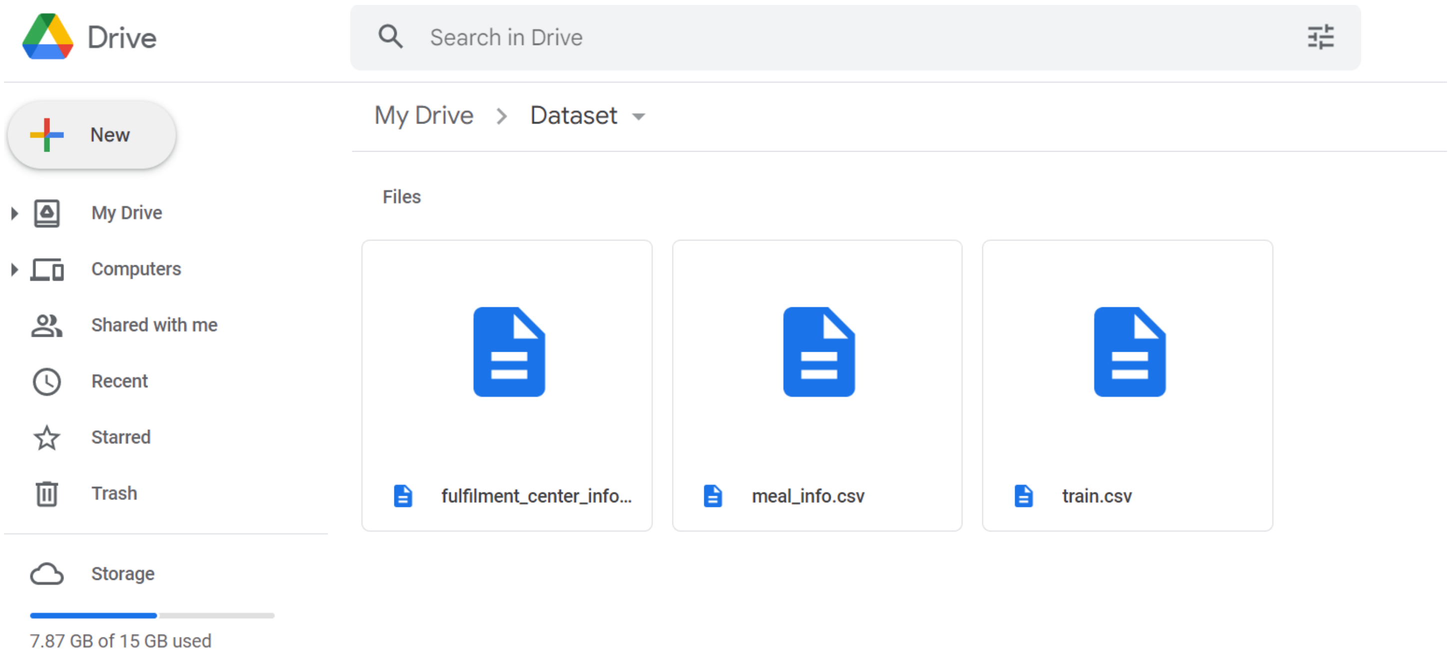

Figure 4: Google Drive

- Step3: The Google Colaboratory(Colab) Environment Setup is for the smooth running of the Python codes and it is very effective when the size of the dataset is huge as it is a cloud-based application. Here, It is configured with my email ID(sasisarath.j@gmail.com) as shown in the figure 5.

3.2 Importing Libraries

During implementation, the necessary libraries are installed and imported for the dataset import, exploratory data analysis, graph plotting, statistical analysis, hyperparameter tuning, model building, and evaluations. The libraries in figure 6 are installed and imported.

3.3 Accessing Data

The data from all three datasets are accessed and combined. We use Google Colab and Jupyter Notebook, so there are two different ways to access the data.

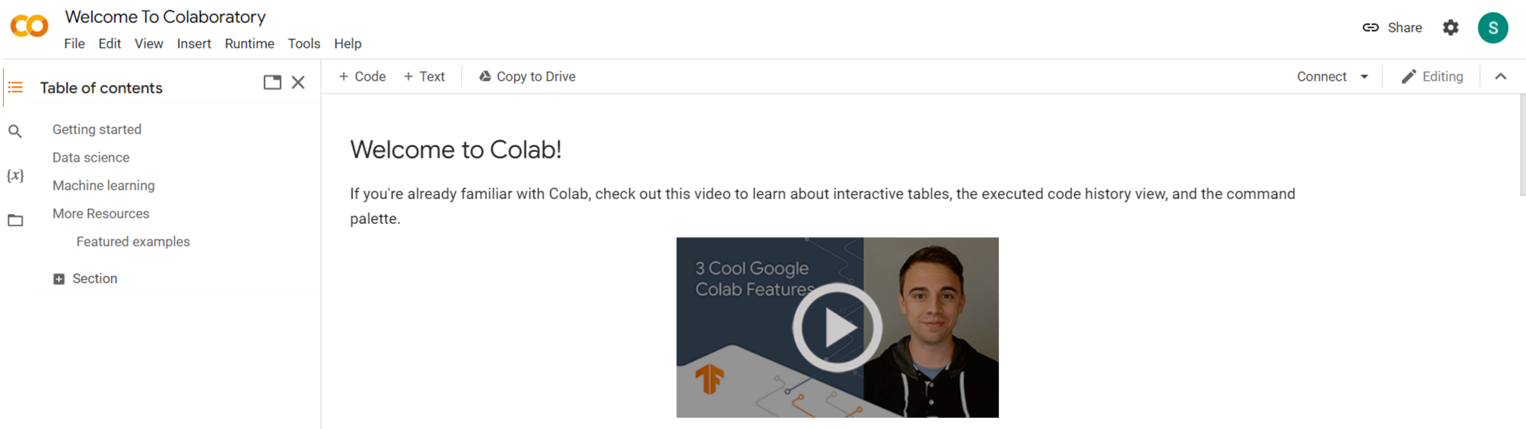

Figure 5: Google Colab

```
In [ ]: #Import Lobraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import time
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBRegressor
from sklearn.metrics import explained_variance_score
from lightgbm import LGBMRegressor
from sklearn.model_selection import GridSearchCV
from catboost import CatBoostRegressor
from sklearn.preprocessing import LabelEncoder
import matplotlib.dates as mdates
from sklearn.svm import SVR
from sklearn.model_selection import train_test_split
import scipy.stats as stats
import seaborn as sns
import matplotlib.pyplot as plt
```

Figure 6: Import Libraries

3.3.1 Accessing from Jupyter Notebook

Figure 7 illustrates the way to access the datasets from the Jupyter Notebook, before that set up a current working directory in Jupyter notebook.

```
In [210]: #Set current working directory
          #os.chdir('D:/Msc Data Analytics/Sem III/Research Project/Dataset')

In [211]: #Importing datasets
          df_food_Orders = pd.read_csv("train.csv")
          df_meal_info = pd.read_csv("meal_info.csv")
          df_center_info = pd.read_csv("fulfilment_center_info.csv", encoding='latin')
```

Figure 7: Jupyter Data Access

3.3.2 Accessing from Google Colab

Figure 8 illustrates the way to access the dataset from the Google drive from the Google Colab.

```
#Mounting google drive
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

Mounted at /content/drive

[ ] #Import datasets
    df_food_Orders = pd.read_csv("/content/drive/MyDrive/Dataset/train.csv")
    df_meal_info = pd.read_csv("/content/drive/MyDrive/Dataset/meal_info.csv")
    df_center_info = pd.read_csv("/content/drive/MyDrive/Dataset/fulfilment_center_info.csv")
```

Figure 8: Colab Data Access

3.3.3 Merging Datasets

As shown in Figure 9, all three datasets are merged.

```
[ ] #Merging all 3 different datasets
    df_food_Orders_fnl = df_food_Orders_fnl.merge(df_meal_info,on='meal_id', how = 'left')
    df_food_Orders_fnl = df_food_Orders_fnl.merge(df_center_info, on='center_id', how = 'left')
```

Figure 9: Merge Dataset

3.4 Missing Value Check

Figure 10 shows that There are no missing values are identified.

```
In [217]: df_food_Orders_fnl.isnull().sum()

Out[217]: id                     0
          Date                   0
          week                   0
          center_id              0
          meal_id                0
          checkout_price         0
          base_price             0
          emailer_for_promotion  0
          homepage_featured      0
          num_orders             0
          year_of_date           0
          month_of_date          0
          month name             0
          category               0
          cuisine                0
          city_code              0
          city_name              0
          region_code            0
          region_name            0
          center_type            0
          op_area                0
          dtype: int64
```

Figure 10: Missing Value

3.5 Data Preprocessing

As shown in figure 11, the following preprocessing steps including missing value check, outlier detection, Log transformation, and deriving new feature variables are carried out during the research work.

Data Preprocessing

```
In [226]: city4={590:'CH1', 526:'CH2',638:'CH3'}
          df_food_Orders_fnl['city_enc_4'] = df_food_Orders_fnl['city_code'].map(city4)
          df_food_Orders_fnl['city_enc_4'] = df_food_Orders_fnl['city_enc_4'].fillna('CH4')

In [227]: # Outlier detection
          plt.figure(figsize= (15,5))
          sns.boxplot(df_food_Orders_fnl['num_orders'])

In [228]: #Remove outliers
          o= df_food_Orders_fnl[df_food_Orders_fnl['num_orders']>15000].index
          df_food_Orders_fnl= df_food_Orders_fnl.drop(o)

In [229]: #Lable encoder
          l=['center_type','op_area','cuisine','category']
          le=LabelEncoder()
          for i in l:
              df_food_Orders_fnl[i]= le.fit_transform(df_food_Orders_fnl[i])

In [231]: #Applying the Log transformation on the target variable
          df_food_Orders_fnl['num_orders'] = np.log(df_food_Orders_fnl['num_orders'])
```

Figure 11: Data Preprocessing

3.6 Feature Scaling and Data Splits

The feature scaling and dataset splits were carried out as shown in figure 12.

```
In [246]: #Feature scores
          abs(df_food_Orders_fnl.corr()['num_orders']).sort_values(ascending=False)

Out[246]: num_orders               1.000000
          checkout_price           0.389111
          base_price               0.329375
          homepage_featured        0.247964
          emailer_for_promotion    0.227161
                                     ...
          center_id_110            0.003348
          meal_id_2640             0.002809
          compare_week_price y/n   0.001519
          id                       0.001379
          center_id_106            0.000152
          Name: num_orders, Length: 155, dtype: float64

In [247]: df_food_Orders_fnl=df_food_Orders_fnl.drop(['id','Date','checkout_price','month_of_date','year_of_date','compare_week_price y/n',

In [248]: X=df_food_Orders_fnl.drop(['num_orders'],axis=1)
          y=df_food_Orders_fnl['num_orders']

In [249]: # train and test split 80% and 20%
          X_train, X_test, y_train, y_test= train_test_split(X, y, test_size = 0.2, random_state=5)
          # train and test split 70% and 30%
          X_train_1, X_test_1, y_train_1, y_test_1= train_test_split(X, y, test_size = 0.3, random_state=5)
```

Figure 12: Feature Scaling and Data Split

3.7 Model Building

Several statistical and machine learning models, including multiple linear regression, lasso, ridge, Bayesian ridge regression, SVR, decision tree, random forest, and gradient boosting regression models, such as Gradian Boosting, XGBoosting, LightGBM, CatBoost, and Facebook Prophet, are used in this research.

Figure 13 illustrates the stages of model construction for multiple linear regression, lasso, and Ridge regression. Additionally, models for Bayesian ridge regression, SVR, decision trees, random forests, and gradian boosting regression, including Gradian Boosting, XGBoosting, LightGBM, and CatBoost, were built. The Facebook prophet model-building steps are illustrated as shown in figure 14.

3.8 Hyperparameter Tuning

The hyperparameter tuning was carried out to find the best possible parameters that improve the performance of the machine learning models. The following Figure 15 illustrates the hyperparameter tuning for the random forest model using the grid search technique. Additionally, the technique is being utilized for other models as well.

3.9 Model Evaluation

The most popular evaluation metrics Chicco D (2021) for regression models are RMSE, MAE, and R^2 are calculated and assessed as shown in figure 16.

Linear Regression

In [251]:
```
final_List=[]
reg = LinearRegression()
start = time.time()
linear_df=reg.fit(X_train, y_train)
Model_Execution_time=round(time.time() - start,2)
#print('Model Execution time: {:.2f}'.format(time.time() - start))
#validate_result(linear_df, 'Linear Regression',)
final_List.append(validate_result(linear_df, 'Linear Regression',Model_Execution_time,X_test,y_test))
```

Lasso and Ridge Regression

In [257]:
```
from sklearn.linear_model import LassoCV
from sklearn.linear_model import RidgeCV

lasso_clf = LassoCV(n_alphas=1, max_iter=3000, random_state=0)
ridge_clf = RidgeCV(gcv_mode='auto')

start = time.time()
lasso_clf_feat = lasso_clf.fit(X_train,y_train)
Model_Execution_time=round(time.time() - start,2)
#validate_result(lasso_clf_feat,'LassoCV')
final_List.append(validate_result(lasso_clf_feat,'LassoCV',Model_Execution_time,X_train,y_train))
#solution_models['LassoCV All feat'] = lasso_clf_feat

start = time.time()
ridge_clf_feat = ridge_clf.fit(X_train,y_train)
Model_Execution_time=round(time.time() - start,2)
#validate_result(ridge_clf_feat,'RidgeCV')
final_List.append(validate_result(ridge_clf_feat,'RidgeCV',Model_Execution_time,X_train,y_train))
#solution_models['RidgeCV All Feat'] = ridge_clf_feat
```

Figure 13: Models

Facebook Prophet

```
#Facebook prophet model building
df = pd.DataFrame()
df['ds'] = pd.to_datetime(df_food_Orders_TimeSeries['Date'])
df['y'] = df_food_Orders_TimeSeries['num_orders']
df.head()
```

[ ]
```
#Setting number of predictions i.e, E.g: 10
prediction_size = 10
train_df = df[:-prediction_size]
test_df=df.tail(prediction_size)
```

[ ]
```
# facebook prophet model building and fitting training data
m = Prophet()
m.fit(train_df)
```

[ ]
```
#Future predictions..
future = m.make_future_dataframe(periods=prediction_size,freq='W')
future.tail(10)
```

[ ]
```
#Forecasting future food orders based on the history
forecast = m.predict(future)
forecast.tail(n=3)
```

Figure 14: Facebook Prophet

```
In [73]: # Hyperparameter Tuning
random_forest_parameters = {
    'n_estimators':[10, 50, 100],
    'max_features':['auto','sqrt','log2'],
    'max_depth':[3, 5, 7],
}

grid_search_RF_feat = GridSearchCV(estimator=random_forest_clf_feat,
                                   param_grid=random_forest_parameters, cv= 5
)
print(grid_search_RF_feat)
grid_search_RF_feat.fit(X_train, y_train)

GridSearchCV(cv=5,
             estimator=RandomForestRegressor(n_estimators=50, random_state=0),
             param_grid={'max_depth': [3, 5, 7],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'n_estimators': [10, 50, 100]})

Out[73]: GridSearchCV(cv=5,
             estimator=RandomForestRegressor(n_estimators=50, random_state=0),
             param_grid={'max_depth': [3, 5, 7],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'n_estimators': [10, 50, 100]})
```

Figure 15: hyperparameter Tuning

```
In [250]: def validate_result(model, model_name,Model_Execution_time,X_test,y_test):
    predicted = model.predict(X_test)
    RSME_score = round(np.sqrt(mean_squared_error(y_test, predicted)),2)
    print('RMSE: ', RSME_score)
    MAE_score = round(mean_absolute_error(y_test, predicted), 3)
    print('MAE: ', MAE_score)
    R2_score = round(r2_score(y_test, predicted),2)
    print('R2 score: ', R2_score)
    return [model_name,RSME_score,MAE_score,R2_score,Model_Execution_time]
```

Figure 16: Evaluation Metrics

	Model Name	RMSE	MAE	R2	Model Training Time
0	Linear Regression	0.64	0.499	0.73	3.35
1	LassoCV	1.04	0.852	0.27	10.73
2	RidgeCV	0.64	0.499	0.73	5.42
3	Bayesian Regression	0.64	0.499	0.73	4.14
4	Decision Tree Regression	0.18	0.062	0.98	7.21
5	Random Forest with All feat	0.25	0.179	0.96	237.18
6	Random Forest with Hyper	0.88	0.716	0.48	85.70
7	Gradient Boosting	0.65	0.511	0.72	127.66
8	XGBoost Regressor	0.45	0.342	0.87	95.77
9	LGB Regressor	0.49	0.380	0.84	3.19
10	CatBoost	0.55	0.424	0.80	497.26

Figure 17: Results for 70:30 splits

	Model Name	RMSE	MAE	R2	Model Training Time
0	Linear Regression	0.64	0.499	0.73	4.55
1	LassoCV	1.04	0.851	0.27	13.62
2	RidgeCV	0.64	0.499	0.73	6.45
3	Bayesian Regression	0.64	0.499	0.73	4.62
4	Decision Tree Regression	0.19	0.067	0.98	8.08
5	Random Forest with All feat	0.26	0.181	0.96	290.79
6	Random Forest with Hyper	0.88	0.716	0.48	99.56
7	Gradient Boosting	0.65	0.510	0.72	159.36
8	XGBoost Regressor	0.45	0.344	0.86	113.13
9	LGB Regressor	0.49	0.379	0.84	3.61
10	CatBoost	0.55	0.424	0.80	605.28

Figure 18: Results for 80:20 splits

4 Appendix

References

Chicco D, Warrens MJ, J. G. (2021). The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation, *Computer Science* .

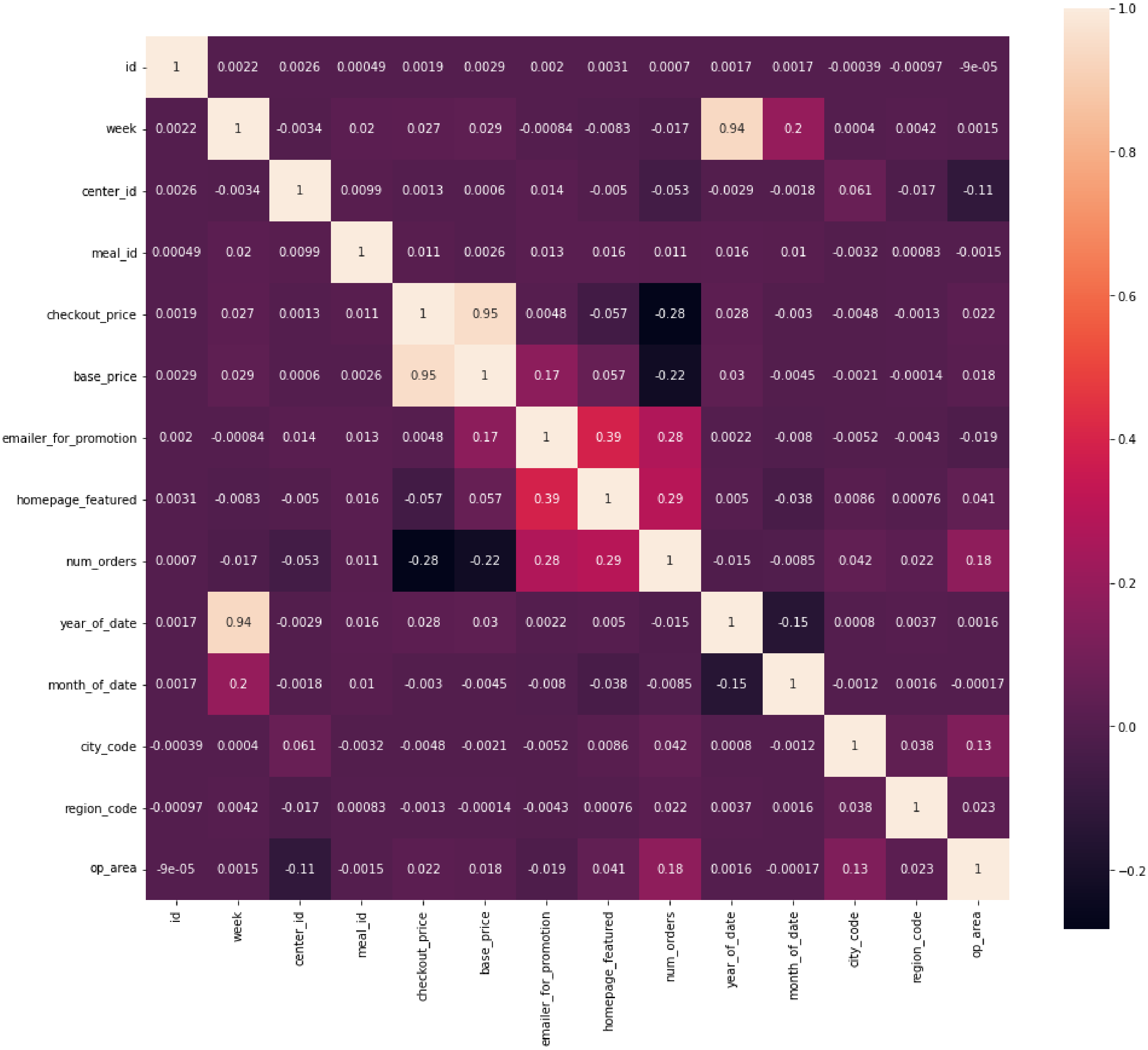

Figure 19: Correlation Matrics