Unit 7: Predicting the Mobility Index in London during the COVID-19 Pandemic

Student's Name: Avinash Bunga

Course: CIS621 Data Analysis for Business Analytics

Professor: Robert Kao

Date: 09.04.2023

Institution: Park University

Predicting the Mobility Index in London during the COVID-19 Pandemic

Introduction

The recent COVID-19 pandemic has considerably reshaped global behaviours, and its nonlinear disruptions have significantly influenced our mobility patterns. In urban areas like London, with lockdowns, curfews, and virus fear, the way we move and interact in public spaces underwent significant shifts.

The dataset we have showcases these shifts. The columns include:
Date: The specific date of data collection.
area_name: The borough in London.
area_code: A unique code representing each borough.

Metrics showing percentage changes from the baseline for:

- ‘re**tail_and_recreation_percent_change_from_baseline’**: Mobility changes in places like restaurants and shopping centres.
- **‘grocery_and_pharmacy_percent_change_from_baseline’:** Changes in visits to grocery markets, food warehouses, farmers markets, speciality food shops, drug stores, and pharmacies.
- **‘parks_percent_change_from_baseline’:** Changes in visits to parks, including public beaches, marinas, dog parks, plazas, and public gardens.
- **‘transit_stations_percent_change_from_baseline’:** Changes at transit stations, such as subway, bus, and train stations.
- **‘workplaces_percent_change_from_baseline’:** Changes in visits to workplaces.

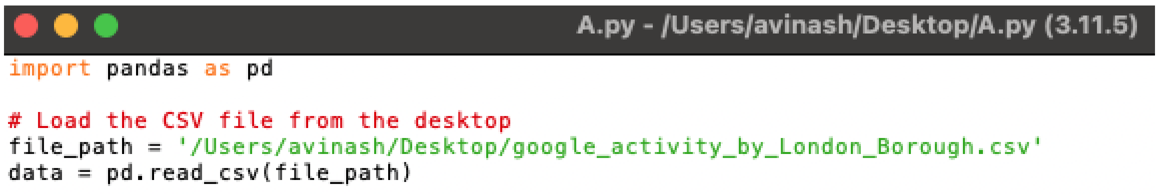
- **‘residential_percent_change_from_baseline’:** Changes in time spent at residences.

Out of these, we've chosen to focus on:

‘retail_and_recreation_percent_change_from_baseline’ column. Why? Because retail and recreational spaces are where most discretionary activities happen – activities that people choose to do, unlike visiting the pharmacy or going to work. By predicting changes in this specific index, we aim to understand how willing people might be to return to 'normal' discretionary behaviours amidst the pandemic.

Methodology

1. Data Loading and Initial Processing



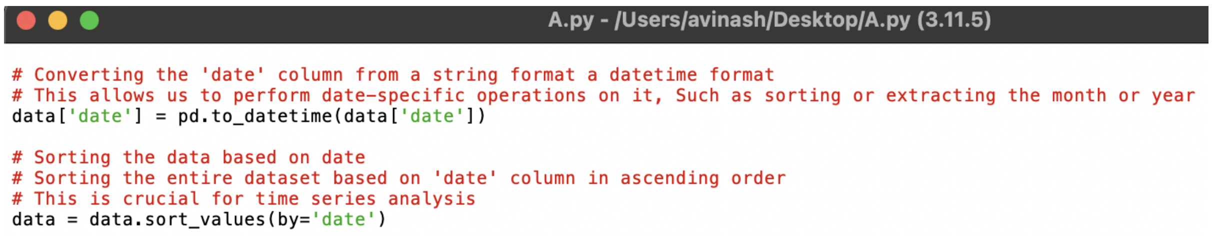
A.py - /Users/avinash/Desktop/A.py (3.11.5)

```python
import pandas as pd

# Load the CSV file from the desktop
file_path = '/Users/avinash/Desktop/google_activity_by_London_Borough.csv'
data = pd.read_csv(file_path)
```

To kickstart our analysis, we import the necessary libraries and the dataset.

2. Date Formatting and Sorting



A.py - /Users/avinash/Desktop/A.py (3.11.5)

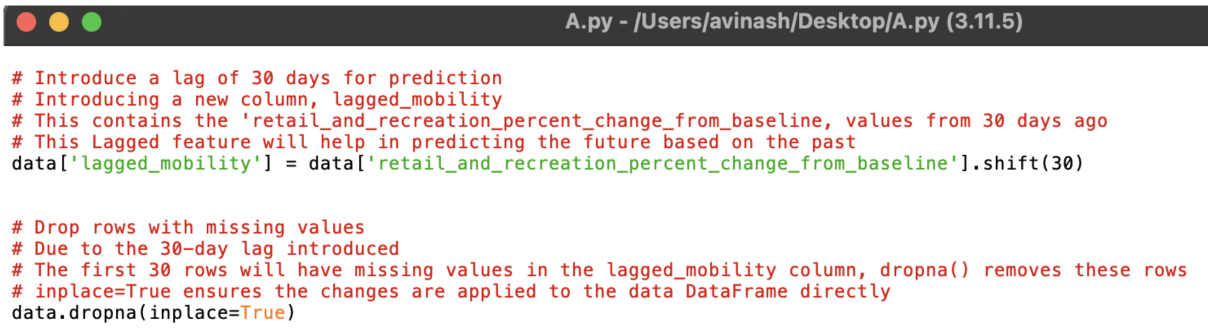
```python
# Converting the 'date' column from a string format a datetime format
# This allows us to perform date-specific operations on it, Such as sorting or extracting the month or year
data['date'] = pd.to_datetime(data['date'])

# Sorting the data based on date
# Sorting the entire dataset based on 'date' column in ascending order
# This is crucial for time series analysis
data = data.sort_values(by='date')
```

Time series data need to be in chronological order. We can work with them more intuitively by converting date strings to date objects.

3. Feature Engineering: Introducing Lag

To accommodate the direction concerning the nonlinearity of COVID-19 disruptions, we chose a 30-day lag. This helps in understanding month-to-month fluctuations but also aids in anticipating sudden changes that might occur due to nonlinear disruptions, such as a sudden lockdown.



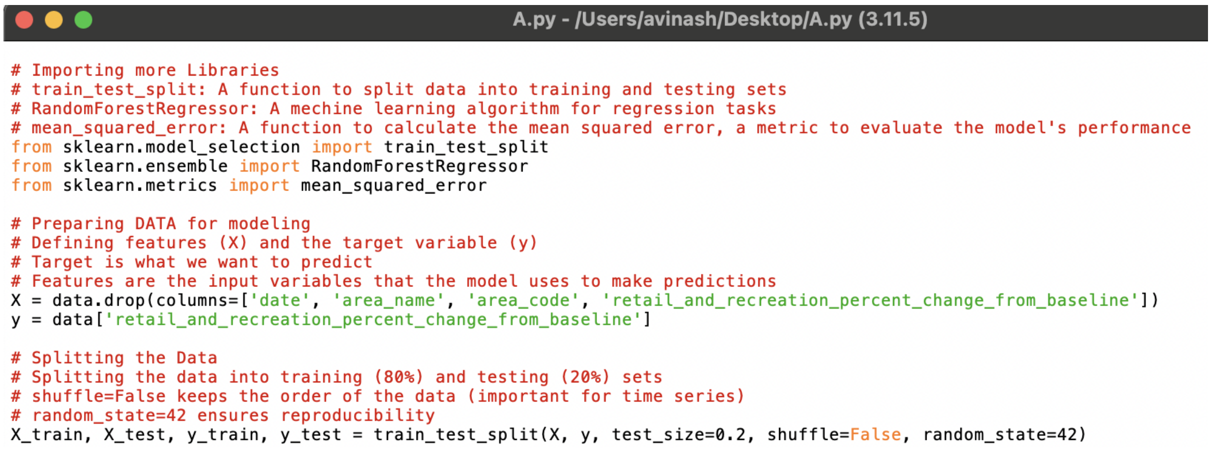
A.py - /Users/avinash/Desktop/A.py (3.11.5)

```
# Introduce a lag of 30 days for prediction
# Introducing a new column, lagged_mobility
# This contains the 'retail_and_recreation_percent_change_from_baseline, values from 30 days ago
# This Lagged feature will help in predicting the future based on the past
data['lagged_mobility'] = data['retail_and_recreation_percent_change_from_baseline'].shift(30)


# Drop rows with missing values
# Due to the 30-day lag introduced
# The first 30 rows will have missing values in the lagged_mobility column, dropna() removes these rows
# inplace=True ensures the changes are applied to the data DataFrame directly
data.dropna(inplace=True)
```

Predicting future mobility changes often requires understanding past behaviours. Introducing a 30-day lag lets us capture the data's temporal nature, making our model informed by the past month's mobility index.

4. Data Preparation



A.py - /Users/avinash/Desktop/A.py (3.11.5)

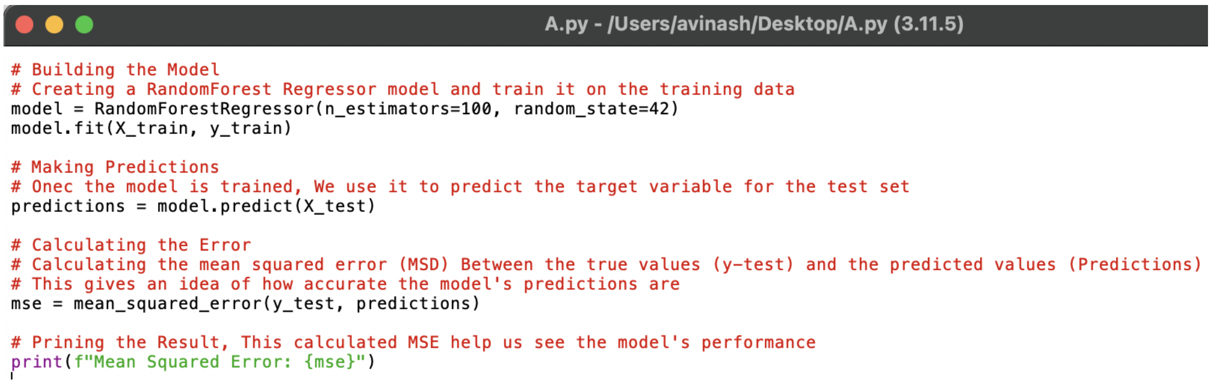
```
# Importing more Libraries
# train_test_split: A function to split data into training and testing sets
# RandomForestRegressor: A mechine learning algorithm for regression tasks
# mean_squared_error: A function to calculate the mean squared error, a metric to evaluate the model's performance
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

# Preparing DATA for modeling
# Defining features (X) and the target variable (y)
# Target is what we want to predict
# Features are the input variables that the model uses to make predictions
X = data.drop(columns=['date', 'area_name', 'area_code', 'retail_and_recreation_percent_change_from_baseline'])
y = data['retail_and_recreation_percent_change_from_baseline']

# Splitting the Data
# Splitting the data into training (80%) and testing (20%) sets
# shuffle=False keeps the order of the data (important for time series)
# random_state=42 ensures reproducibility
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False, random_state=42)
```

We're now setting the stage for modelling. We separate our features (X) from our target (y) and split our dataset into training and testing sets.

5. Model Building & Evaluation



```python
# Building the Model
# Creating a RandomForest Regressor model and train it on the training data
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Making Predictions
# Onec the model is trained, We use it to predict the target variable for the test set
predictions = model.predict(X_test)

# Calculating the Error
# Calculating the mean squared error (MSD) Between the true values (y-test) and the predicted values (Predictions)
# This gives an idea of how accurate the model's predictions are
mse = mean_squared_error(y_test, predictions)

# Prining the Result, This calculated MSE help us see the model's performance
print(f"Mean Squared Error: {mse}")
```

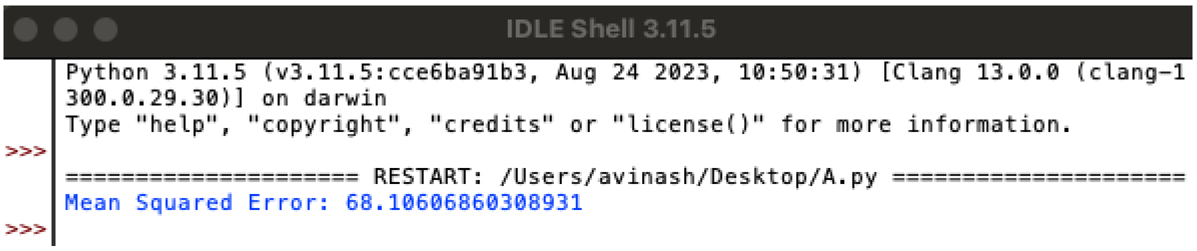
For building the model, we leaned towards a RandomForestRegressor, a non-linear model that can handle and adapt to the non-linear disruptions of COVID-19. This choice was made considering the unpredictable ebb and flow of the pandemic and its varying impact on mobility.

A RandomForestRegressor helps predict the future mobility index. The mean squared error tells us how close our predictions are to the actual values.

Executing the Script and Observing the Results

After carefully scripting and ensuring our data preprocessing was sound, we executed our Python script. This script was run in the IDLE shell, the integrated development environment for Python. The outcome of our RandomForestRegressor model was a prediction of the future mobility index based on past data.

Result in the IDLE Shell:



```
    Python 3.11.5 (v3.11.5:cce6ba91b3, Aug 24 2023, 10:50:31) [Clang 13.0.0 (clang-1
    300.0.29.30)] on darwin
    Type "help", "copyright", "credits" or "license()" for more information.
>>> 
    ====================== RESTART: /Users/avinash/Desktop/A.py ======================
    Mean Squared Error: 68.10606860308931
>>>
```

Discussion on Model Limitations, Predictive Power, and the Role of P-value

When examining our model's performance and the underlying methodology, it's essential to understand its inherent limitations and the role of statistical significance in shaping our conclusions.

Model Limitations:

Our RandomForestRegressor, while robust in handling non-linearity and intricate interactions in data, might sometimes anticipate sudden and unprecedented changes correctly. This limitation emerges from the model's reliance on past data. For instance, the model might not foresee its implications on mobility if there's a sudden governmental decision or an unexpected public event.

Predictive modelling, especially with dynamic real-world data like ours, faces the challenge of "concept drift," where the target variable's statistical characteristics fluctuate over time. The patterns our model learns from the past might only sometimes be valid for the future, especially in rapidly changing scenarios like a pandemic.

Statistical Significance vs. Predictive Power:

Statistical significance, often denoted by a p-value in regression models, tells us if a variable has a significant relationship with the outcome variable. However, a low p-value (indicative of statistical significance) sometimes equates to practical or meaningful significance.

In the context of our model, even if specific predictors have a statistically significant relationship with the mobility index, it doesn't guarantee that the model will have a strong predictive power for unseen data. Predictive accuracy – the ability of our model to forecast future data points correctly – is more vital than fitting the training data perfectly.

While we've used RandomForest (which doesn't provide p-values like traditional regression models), the point remains: ensuring a model generalises well to new data is paramount.

Improving Model Quality:

To enhance our model's quality and predictive power, we might consider:

Integrating more dynamic data sources: Incorporating real-time data such as public sentiment on mobility, fresh updates on COVID-19 cases, or vaccination rates can provide more current context to the model.

More granular data: Data at a neighbourhood or street level, instead of just boroughs, can offer finer insights and detect localised patterns.

External datasets: Information on local events, governmental policies, and public transportation schedules might help anticipate sudden shifts in mobility patterns.

Conclusion

Our journey through the dataset has been insightful. By focusing on the '**retail_and_recreation_percent_change_from_baseline**' column, we've tried to capture the essence of people's willingness to engage in discretionary activities during the pandemic. The RandomForestRegressor, an ensemble of decision trees, has provided predictions for the future mobility index.

Our model yielded a **Mean Squared Error (MSE) of approximately 68.11.** This error offers an insight into the model's accuracy. The MSE is a measure that tells us how close our predictions are to the actual values. A lower MSE indicates better accuracy, but it's essential to understand the context of this number. Given the vast number of data points and the diverse range of mobility indices, an MSE of 68.11 can be considered reasonable for our current model. However, continuous refinement is key to improving accuracy further.

In predictive modelling, it's essential to remember that models simplify reality. While our model gives predictions based on past data, the real world is more dynamic. Policies change, new strains of the virus might emerge, and people's behaviours adapt. Thus, while our model is a valuable tool, it is one piece of the larger puzzle of understanding and predicting human behaviour in these unprecedented times.

By consistently refining our model and integrating more dynamic and up-to-date datasets, We can improve our forecasts and provide more insightful analyses about how mobility will develop in London and, potentially, around the globe.

Data Source: Google Mobility by Borough - Link

References:

Binieli, M. (2018, October 16). ***Machine learning: An introduction to mean squared error and regression lines.*** **Freecodecamp. https://www.freecodecamp.org/news/machine-learning-mean-squared-error-regression-line-c7dde9a26b93/**

V. (2021, January 3). ***How to Calculate Mean Squared Error (MSE) in Python*****. Vedexcel. https://vedexcel.com/how-to-calculate-mean-squared-error-mse-in-python/**

Bedre, R. (2022, September 25). ***How to perform one and two-sample t-test in Python*****. Reneshbedre. https://www.reneshbedre.com/blog/ttest.html**