Group 13

Dataset on laundry shop

Chan Yun Hong	1181103263
Boe Chang Horn	1181103320
Tan Kai Yuan	1181103087

Introduction

- **Main Question 1**: What factors contribute to the change in the total money spent by customers?
  - Focus on the factors that can increase the total money spent by customers,
  - Uplifting the monthly revenue
- **Main Question 2**: What are the frequent types of customers who like to do laundry for blankets or clothes?
  - Identify customer cluster
  - Create targeted campaign

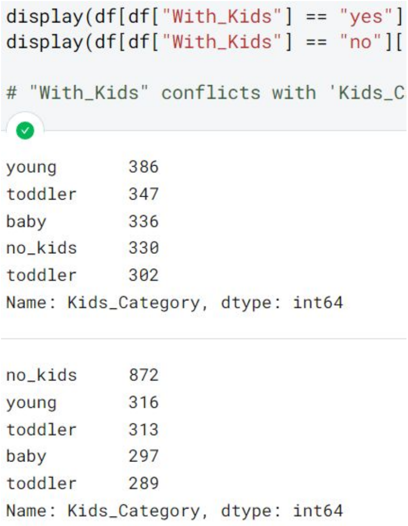
Exploratory Data Analysis

External Dataset

- Location dataset from GeoPy’s OpenStreetMap API
  - City, State, and Postcode
- Weather dataset from Visual Crossing’s Historical Weather Data API
  - Temperature, Precipitation, Humidity, Weather Condition, etc.

Data Preprocessing

- Conflict and redundant feature such as “Kids_Category” and “With_Kids”
    - Keep “Kids_Category” and drop “With_Kids”
    - Use “With_Kids” to fill “Kids_Category”’s missing value
- Value with format issue
    - Additional space such as “foreigner ” vs “foreigner”
- Data Transformation
    - “preciptype” to “isRain”
    - “DayOfWeek” and “HourInDay” from “Datetime”



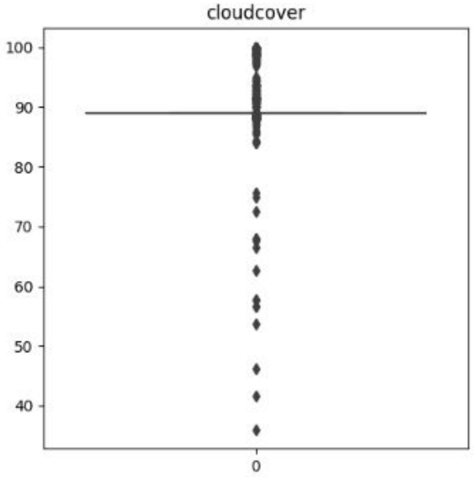
```
display(df[df["With_Kids"] == "yes"]
display(df[df["With_Kids"] == "no"][

# "With_Kids" conflicts with 'Kids_C
```

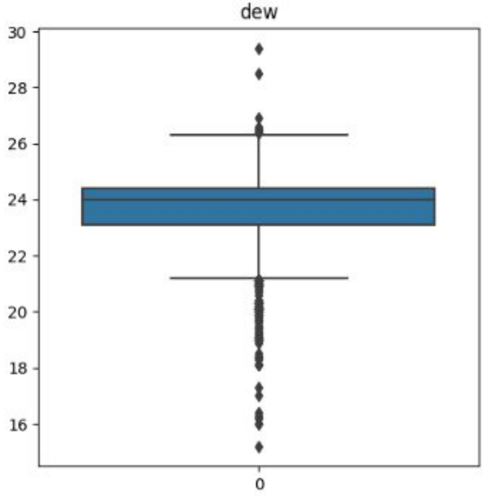
```
young       386
toddler     347
baby        336
no_kids     330
toddler     302
Name: Kids_Category, dtype: int64
```

```
no_kids     872
young       316
toddler     313
baby        297
toddler     289
Name: Kids_Category, dtype: int64
```

Outliers and Missing Values

- Missing Values
  - Drop features with high missing value and unnecessary features
  - Drop left over rows with missing value
- Outliers
  - Plot boxplots to see data distribution and outliers
  - Drop the columns that have extremely high outliers
  - Use machine learning model that is robust for dealing with outliers



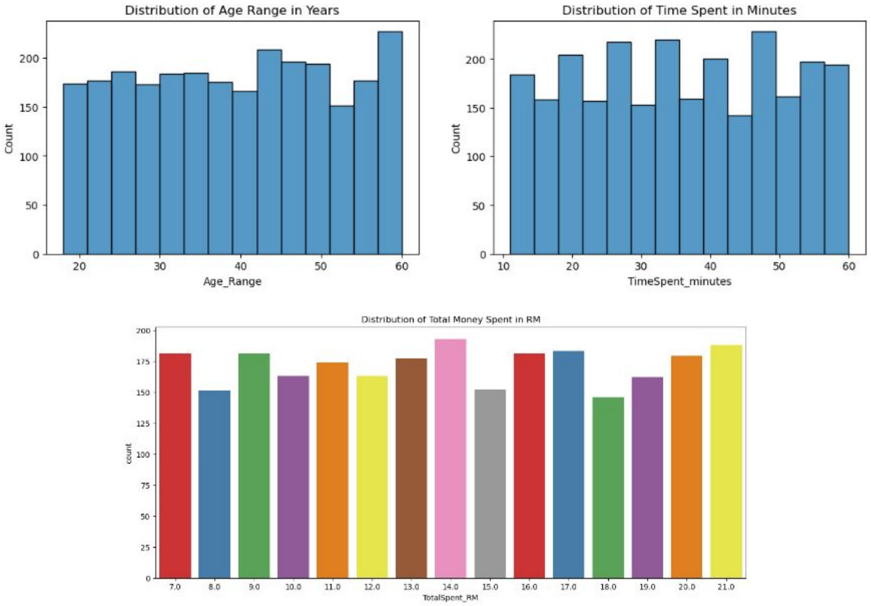


Exploratory Data Analysis

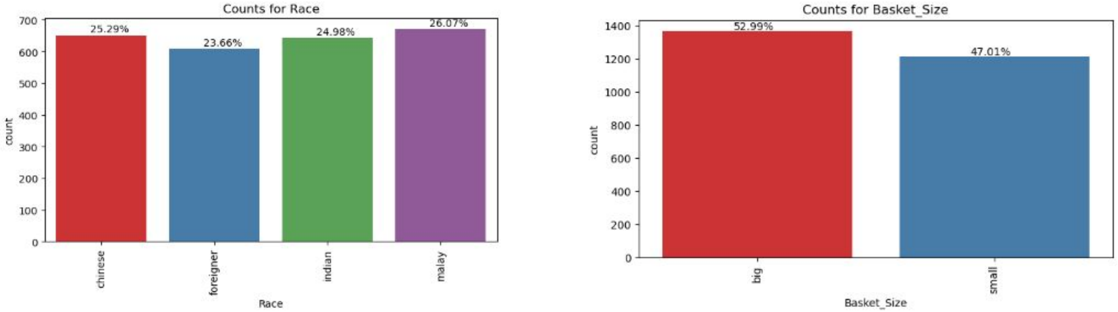
Univariate Analysis

- Draw barcharts for **Categorical** and **Numerical** data as **Data Visualization**
- Each column is distributed fairly balance
- Distribution of the data is fairly equal
- **No data imbalance treatment is necessary**

Data Visualization for Univariate Analysis of Categorical Variables



Data Visualization for Univariate Analysis of Numerical Variables

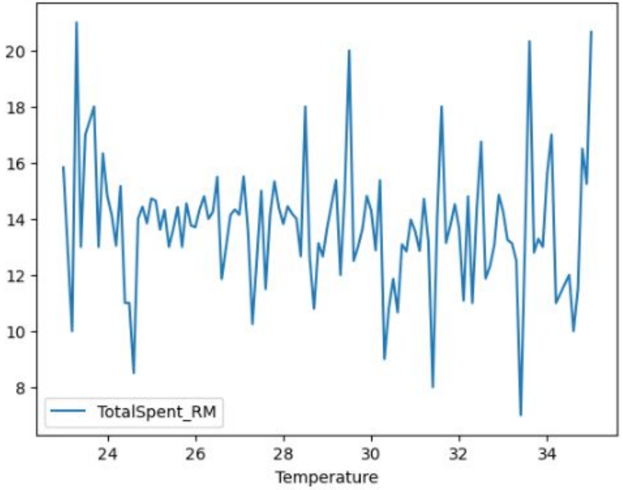


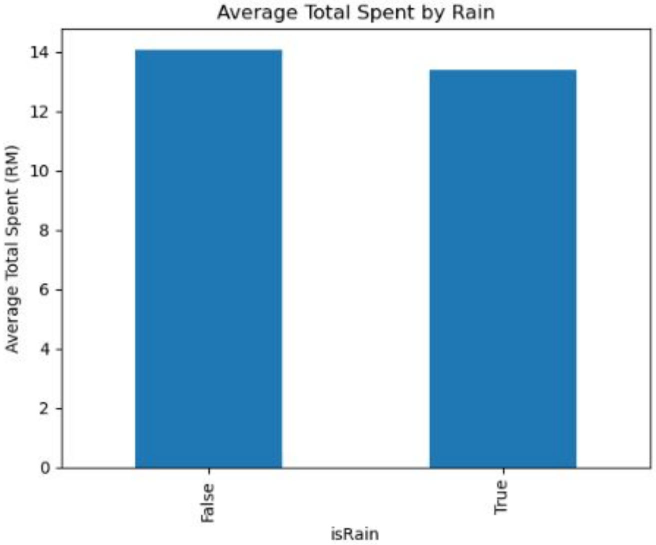
Relationship between Variables

“How does the weather affect the total money spent on average?”

- Line plot (Temp VS Total Spend)
- No significant difference found
- Line is fluctuating throughout the x-axis

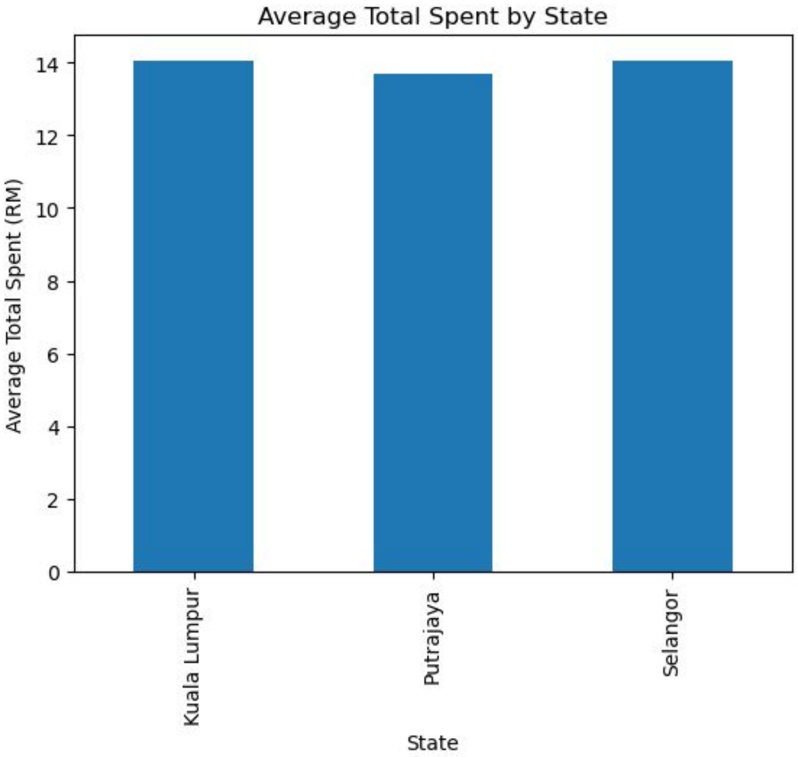
- Barplot (Rain VS Total Spend)
- Distribution is reasonably balanced
- No significant difference found





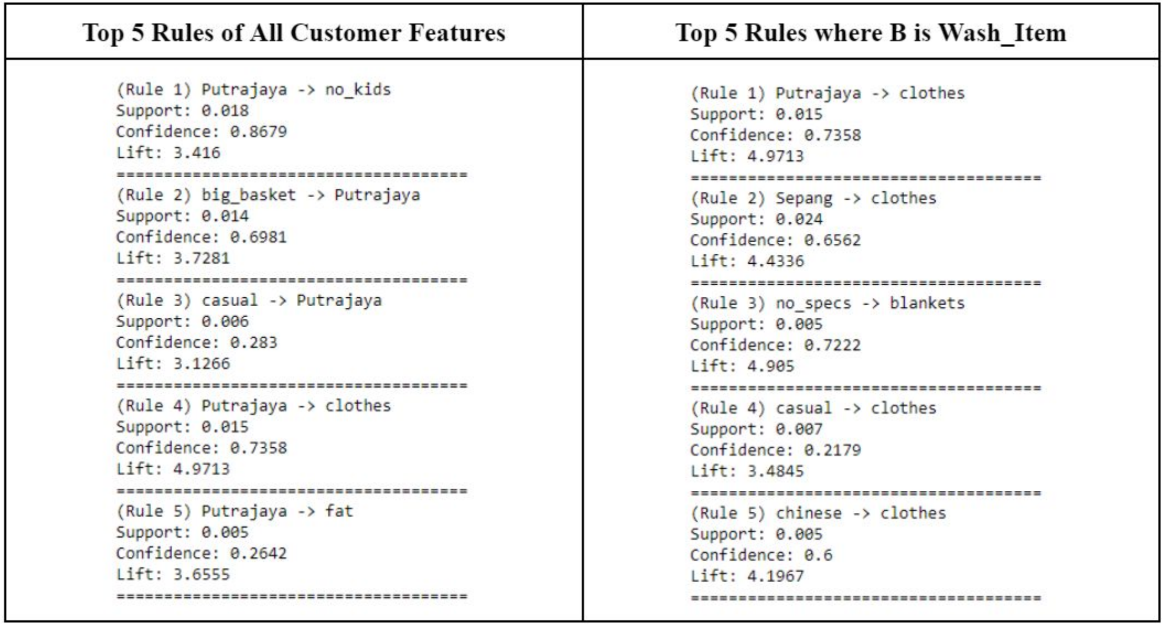
“How does the location of customer affect the total money spent?”

- Plot barplot between State and Average Total Spend
- No significant difference found
- Distribution of the Average Total Spend for each State is fairly equal



Association Rule Mining

- “What are the frequent types of customers who like to do laundry for blankets or clothes?”
- **Lift(A -> B)** , where A is the feature of customer (race, shirt_type and etc) and B is wash item (clothes or blankets)
- Display Top 5 Rules (All Customer Features & B = Wash Item)



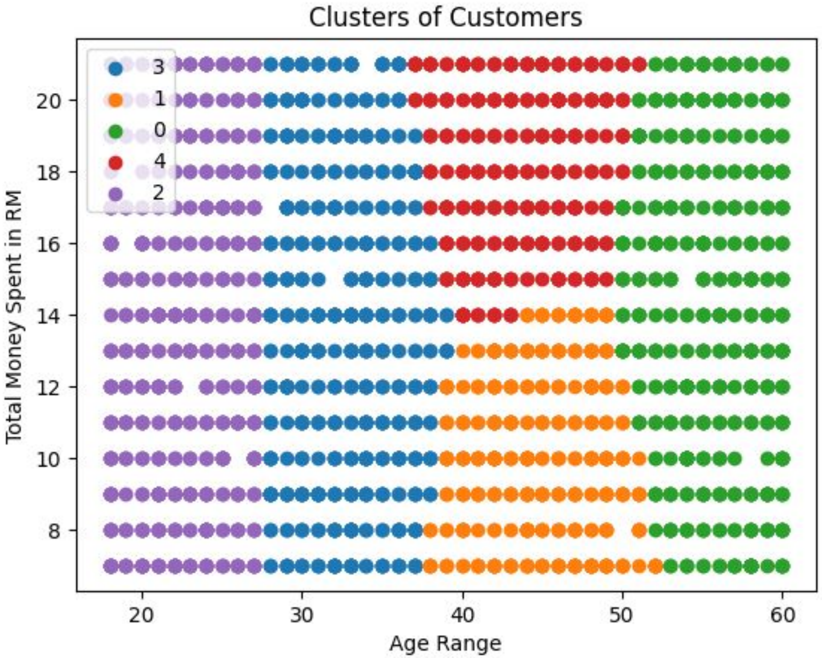
Top 5 Rules of All Customer Features	Top 5 Rules where B is Wash_Item
(Rule 1) Putrajaya -> no_kids Support: 0.018 Confidence: 0.8679 Lift: 3.416 ====================================== (Rule 2) big_basket -> Putrajaya Support: 0.014 Confidence: 0.6981 Lift: 3.7281 ====================================== (Rule 3) casual -> Putrajaya Support: 0.006 Confidence: 0.283 Lift: 3.1266 ====================================== (Rule 4) Putrajaya -> clothes Support: 0.015 Confidence: 0.7358 Lift: 4.9713 ====================================== (Rule 5) Putrajaya -> fat Support: 0.005 Confidence: 0.2642 Lift: 3.6555 ======================================	(Rule 1) Putrajaya -> clothes Support: 0.015 Confidence: 0.7358 Lift: 4.9713 ====================================== (Rule 2) Sepang -> clothes Support: 0.024 Confidence: 0.6562 Lift: 4.4336 ====================================== (Rule 3) no_specs -> blankets Support: 0.005 Confidence: 0.7222 Lift: 4.905 ====================================== (Rule 4) casual -> clothes Support: 0.007 Confidence: 0.2179 Lift: 3.4845 ====================================== (Rule 5) chinese -> clothes Support: 0.005 Confidence: 0.6 Lift: 4.1967 ======================================

- Rule 1 Example :
  - 1.5% of Putrajaya customers do laundry for clothes (Support)
  - 4.9 times more likely (Lift)

- Customers no glasses have the highest likelihood of washing blankets (Lift = 4.905).
- Customers who live in Putrajaya have the highest chance of washing clothes (Lift = 4.9713).

Clustering

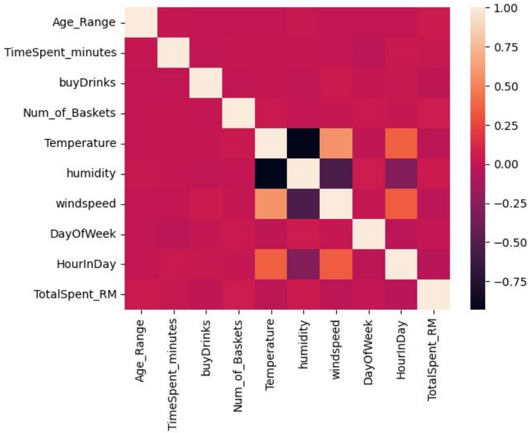
- Use K-means clustering
- Only includes age range and total money spent variable
- Findings
  - Red customer cluster
    - Average age range is 43
    - Contribute more sales to laundry shop
  - Orange customer cluster
    - Average age range is 45
    - Contribute fewer sales to laundry shop

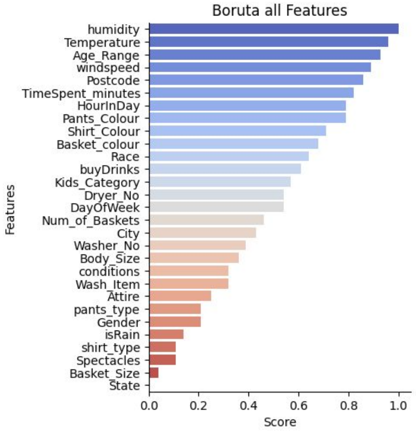


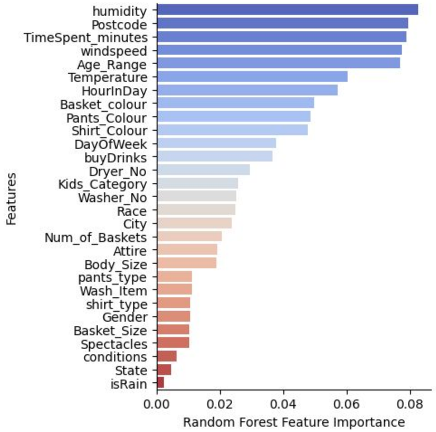
Feature Selection

Type of Feature Selection Methods

1. Target variable: Total spent by customer
2. Filter Method
   a. Pearson Correlation
      i. No strong correlation between dependent variable and independent variable
3. Wrapper Method
   a. Boruta algorithm
4. Embedded Method
   a. Random Forest feature importance

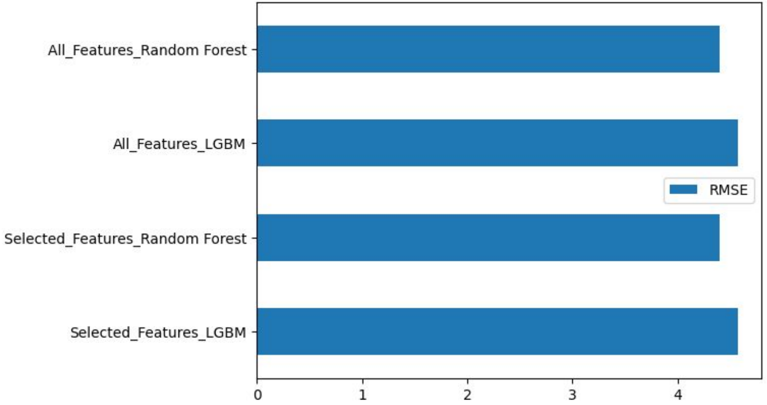






Proposed methodology to finalize features

1. Get the set of features that have the best performance from both feature selection techniques which are the Boruta algorithm and Random Forest feature importance, then
2. Retrieve the overlapped features of both sets of features as selected features.
3. Compare the performance between the model built using selected features and the model built using all available features to determine the final set of features



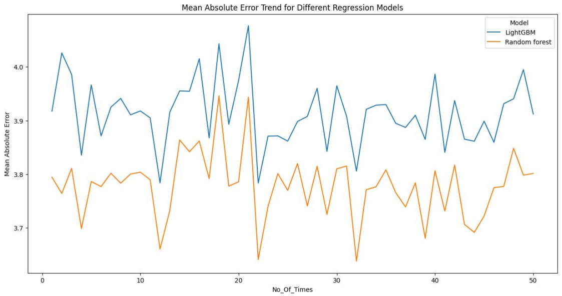
Model Construction and Comparison

Regression

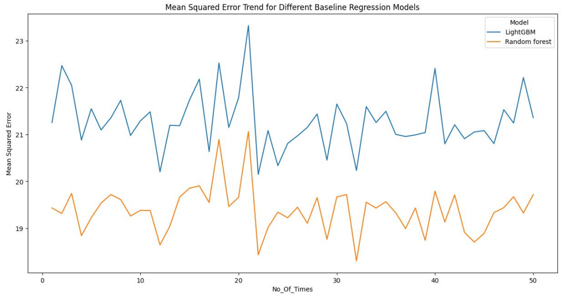
- Chosen Models
  - Random Forest
    - Usually has decent performance
    - Robust to overfitting
  - LightGBM
    - Faster training speed
    - Higher efficiency
    - Lower memory usage
    - Better accuracy
- Model Validation
  - Train-test split
    - See how well the model generalize on unseen data
  - K-fold cross validation
    - Minimize sampling bias

Regression

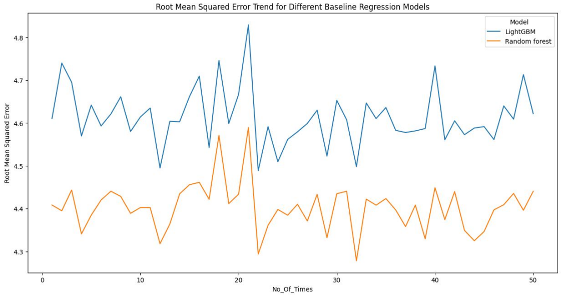
- Visualize the output of the model
  - Plotting the trend of RMSE of baseline models in multiple iterations
- Model comparison
  - By checking the average difference of model metrics
    - Random Forest is better
  - Running statistical test (Wilcoxon Test)
    - Using 15 fold cross validation to generate 15 pairs of RMSE from both type of model
    - Result
      - P-value is lower than 0.05, indicating that there is statistically significant difference between the RMSE
      - Random Forest is better



Mean Absolute Error Trend for Different Regression Models



Mean Squared Error Trend for Different Baseline Regression Models



Root Mean Squared Error Trend for Different Baseline Regression Models

Regression

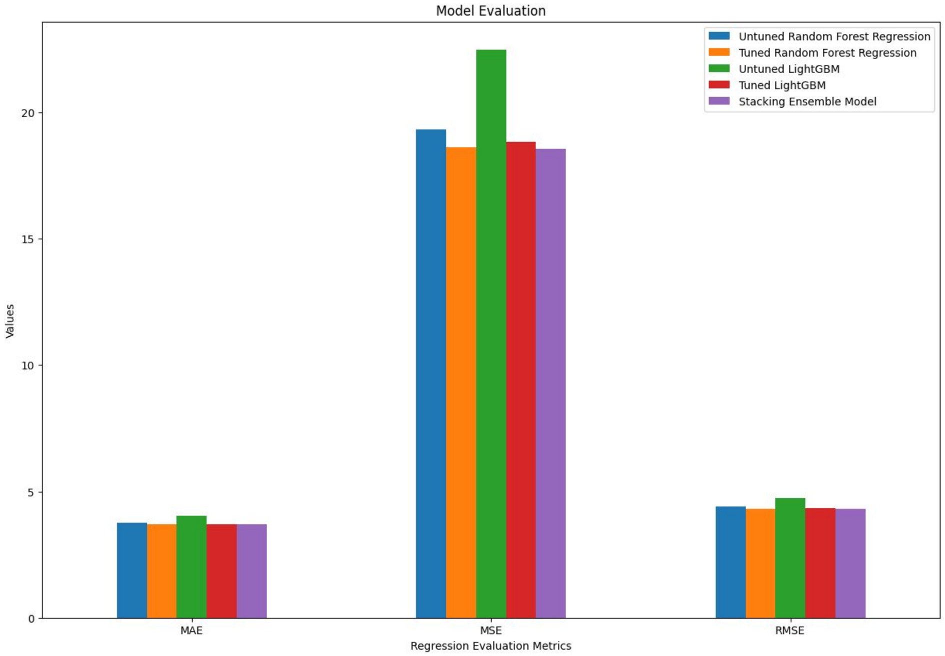
- Impact of SMOTE and non-SMOTE datasets on the model performance
  - Generating SMOTE dataset
  - Using 15 fold cross validation to generate 15 pairs of RMSE from model that use SMOTE dataset and model that use non-SMOTE dataset
  - Result
    - P-value is lower than 0.05, indicating that there is statistically significant difference between the RMSE
    - SMOTE dataset does improve the model performance

Regression

- Hyperparameter tuning
  - Certain model parameters are included such as number of boosted trees to fit for Random Forest
  - Numeric range of the parameters values are relatively small
  - Performed using Grid Search algorithm

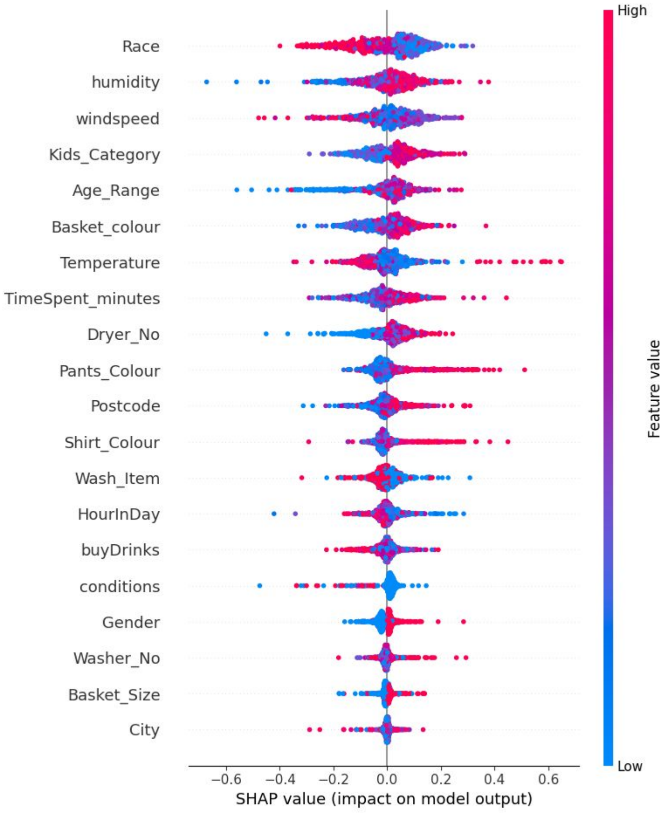
Regression

- Model Evaluation between all models
  - Metrics used
    - Mean Absolute Error (MAE)
    - Mean Squared Error (MSE)
    - Root Mean Squared Error (RMSE)
  - Stacking ensemble model performs the best



Regression

- Interpret how the features impact the model prediction by
  - using SHAP values
- For example
  - High values of humidity will have a positive contribution to the prediction



Classification

- Chosen Models
  - Random Forest Classifier
    - Excels at handling overfitting
    - Strong emphasis on features
  - Naive Bayes
    - Straightforward
    - Efficient
    - Well-suited for approximate each category with the given features
    - Produce examples with comparable characteristics

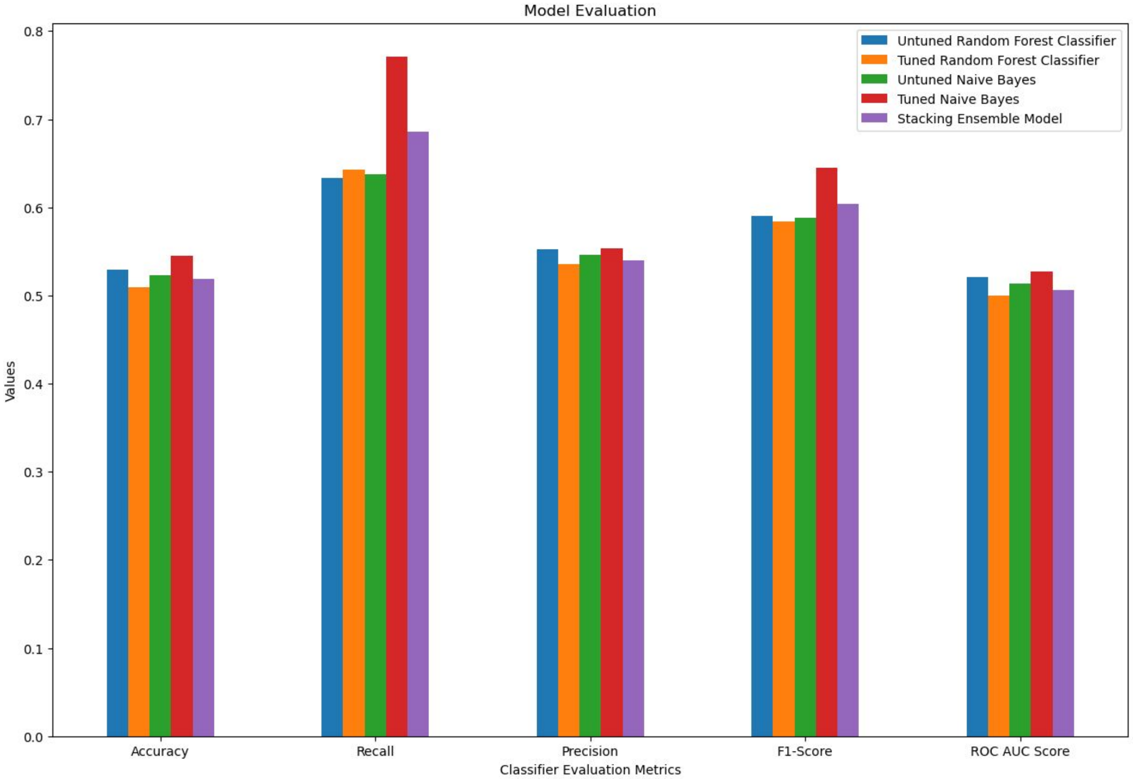
Classification

Hyperparameter tuning

- Naive Bayes
    - var smoothing - regulate the level of regularization used
    - help avoid overfitting
- Random Forest Classifier
    - Certain model parameters are included such as number of boosted trees to fit for Random Forest
    - Numeric range of the parameters values are relatively small
- Performed using Grid Search algorithm

Classification

- Model Evaluation between all models
  - Metrics used
    - Accuracy
    - Recall
    - Precision
    - F1-Score
    - ROC AUC Score
  - Tuned Naive Bayes model performs the best

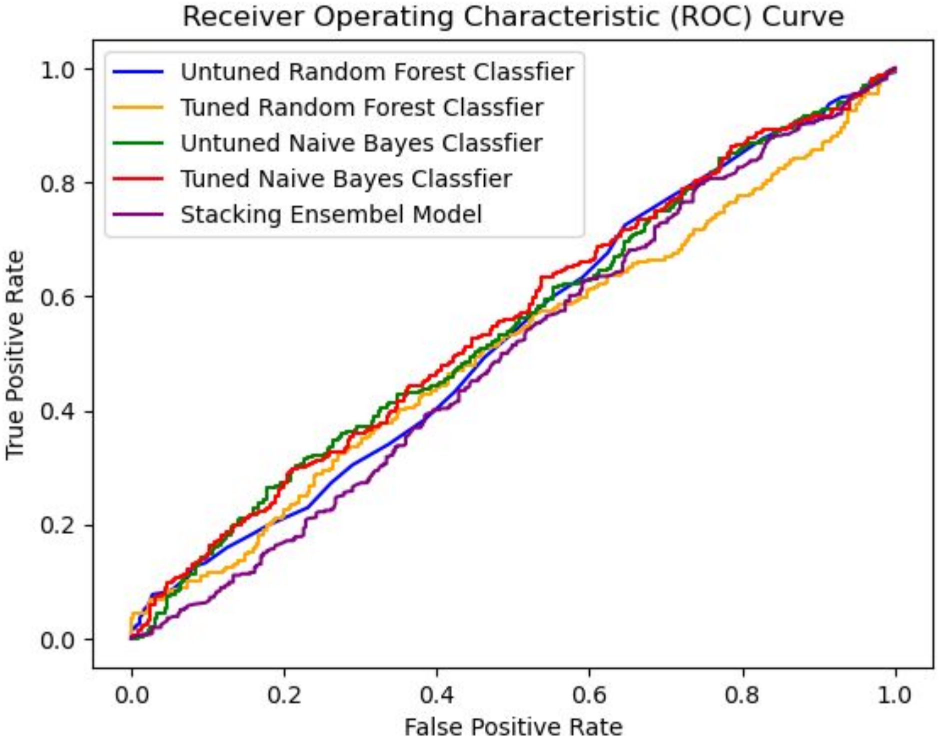


Classification

- Plot ROC curve
  - Discriminate between positive and negative classes

- All models performed quite similarly
  - Very close to random guessing
  - Approaching the diagonal line.

- Tuned Naive Bayes model produced slightly better results than the other 4 models.



Improve the performance of both classification and regression models

- Cause of low performance
  - The current dataset is in low quality, proved by
    - Pearson correlation
    - Data is not consistent between columns
- Solution
  - Apply data-centric strategy
  - Ensure the data is collected in a systematic and discrete way
  - Enhance data quality

Deployment

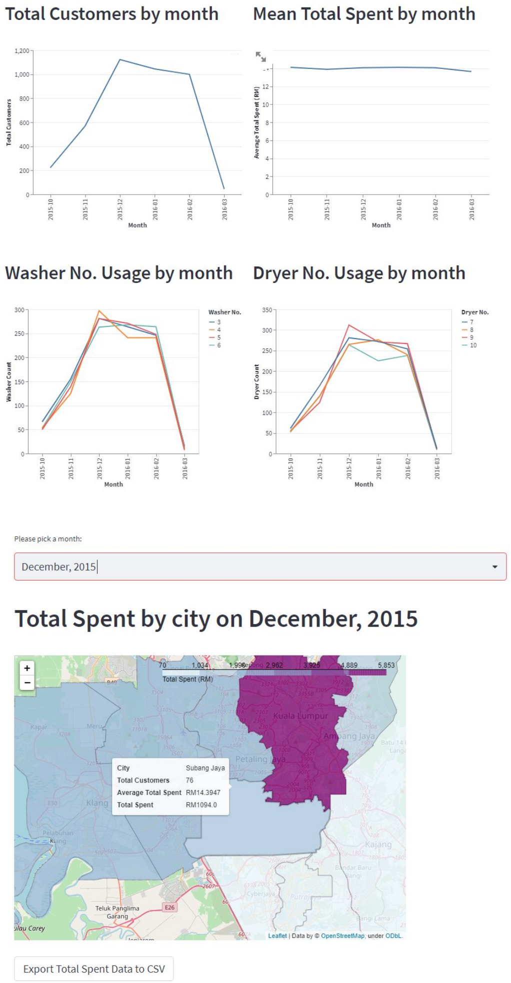
Real-time Visualization

Line plots

- Total Customer by month
- Mean Total Spent by month
- Washer No. Usage by month
- Dryer No. Usage by month

Interactive Choropleth Map

- Identify location of customer base
- Total Customers, Average Total Spent, Total Spent
- Monthly analysis
- Data export to CSV for further analysis



Total Customers by month

Mean Total Spent by month

Washer No. Usage by month

Dryer No. Usage by month

Please pick a month:

December, 2015

Total Spent by city on December, 2015

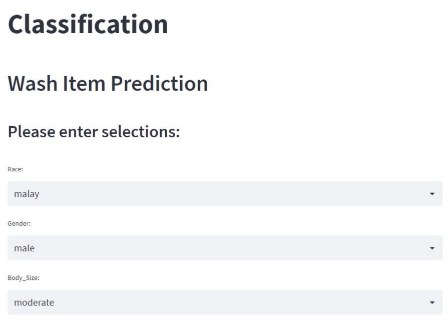
City Subang Jaya
Total Customers 76
Average Total Spent RM14.3947
Total Spent RM1094.0

Export Total Spent Data to CSV

Real-time Prediction

- Regression model
    - Predict Total Spent based on various attributes
    - Real-time prediction
    - Easy to use

- Classification model
    - Predict Wash Item based on various attributes
    - Instantaneous prediction
    - Input selections to predict



Classification

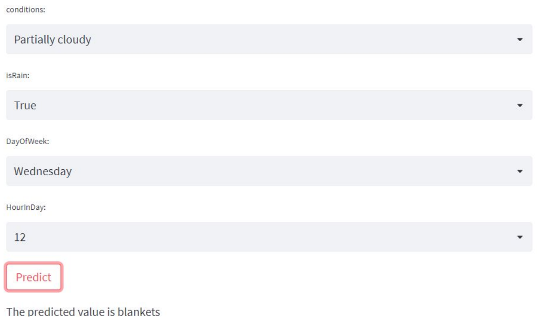
Wash Item Prediction

Please enter selections:

Race: malay

Gender: male

Body_Size: moderate



conditions: Partially cloudy

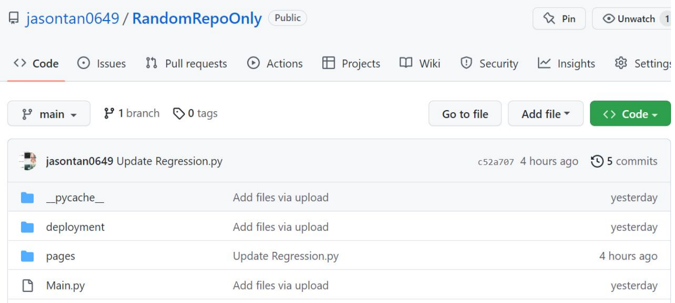
isRain: True

DayOfWeek: Wednesday

HourInDay: 12

The predicted value is blankets

Streamlit Cloud Hosting



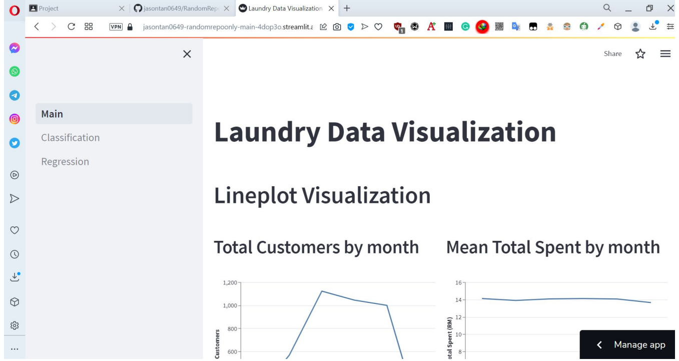
jasontan0649 / RandomRepoOnly Public

1 branch 0 tags

jasontan0649 Update Regression.py c52a707 4 hours ago 5 commits

__pycache__	Add files via upload	yesterday
deployment	Add files via upload	yesterday
pages	Update Regression.py	4 hours ago
Main.py	Add files via upload	yesterday

- Source code published as a GitHub repository
- Hosted on Streamlit Cloud server
- Real-time data visualization and prediction everywhere
- URL: https://jasontan0649-randomrepoonly-main-4dop3o.streamlit.app



Laundry Data Visualization

Lineplot Visualization

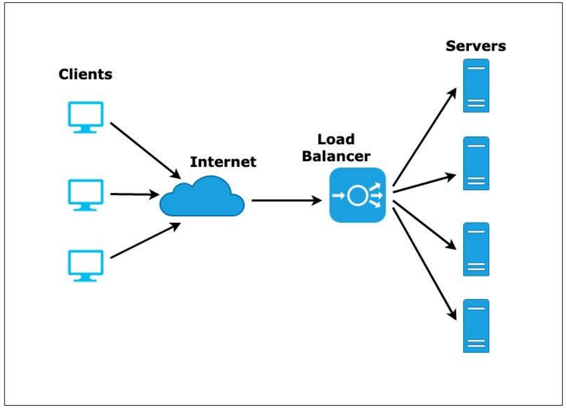
Total Customers by month

Mean Total Spent by month

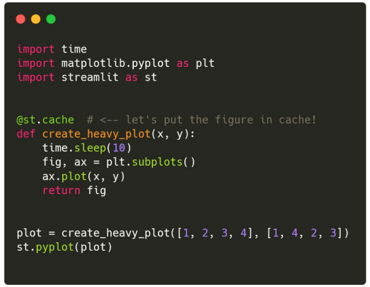
Performance Improvement

- Load Balancing Mechanism
  - Ensure high availability

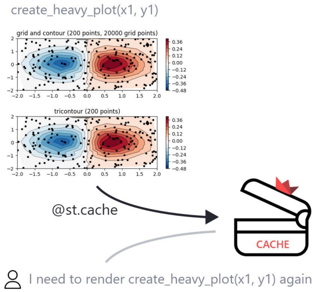
- Caching
  - Reduce usage of computing resources



You want to cache your awesome figures to speed up the render



```
import time
import matplotlib.pyplot as plt
import streamlit as st


@st.cache  # <-- let's put the figure in cache!
def create_heavy_plot(x, y):
    time.sleep(10)
    fig, ax = plt.subplots()
    ax.plot(x, y)
    return fig


plot = create_heavy_plot([1, 2, 3, 4], [1, 4, 2, 3])
st.pyplot(plot)
```


create_heavy_plot(x1, y1)
grid and contour (200 points, 20000 grid points)
tricontour (200 points)
@st.cache
CACHE
I need to render create_heavy_plot(x1, y1) again