Project

<u>Mission 1</u>

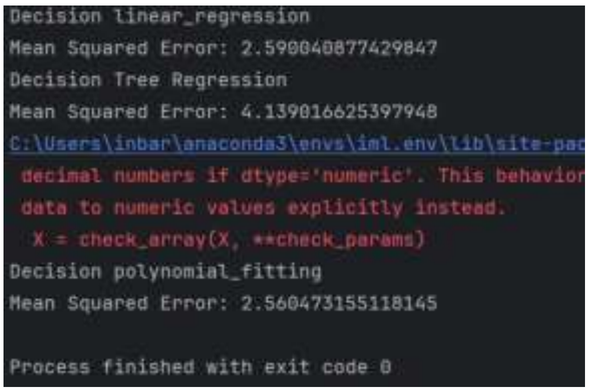
<u>Libraries Used:</u>

- pandas, numpy, matplotlib ,xgboost, sklearn, os, logging, argparse.

<u>Baseline</u>

1. <u>Data Preprocessing</u>
   - **Datetime Conversion:** Key time-based columns (arrival_time, door_closing_time) are converted to datetime objects.
   - **Door Delta Calculation:** The time difference between door closing and arrival times is calculated and used as a feature.
   - **Categorizing Arrival Times:** Arrival times are categorized into bins based on percentiles to capture the time-of-day effect.
   - **Label Encoding:** Categorical features (part, alternative, cluster) are encoded numerically for model compatibility.
   - **Handling Missing Values**: Missing values in key columns are handled by filling with the mean.
2. <u>Data Splitting:</u>
   - For the baseline we used 5% of the total data. This data is split into training and testing sets.
3. <u>Modeling</u>

   we tried to use 3 different modeling:
   1. Decision Trees
   2. Linear Regression - helps the linear relationship between features and the target variable. We also did feature evaluations.
   3. Polynomial Regression – We utilized a second-degree polynomial regression model. We tried a few degrees – 1,2,3,4.



```
Decision linear_regression
Mean Squared Error: 2.590040877429847
Decision Tree Regression
Mean Squared Error: 4.139016625397948
C:\Users\inbar\anaconda3\envs\iml.env\lib\site-pac
 decimal numbers if dtype='numeric'. This behavior
 data to numeric values explicitly instead.
  X = check_array(X, **check_params)
Decision polynomial_fitting
Mean Squared Error: 2.560473155118145

Process finished with exit code 0
```

Eventually, we selected Polynomial Regression due to its lowest mean squared error.

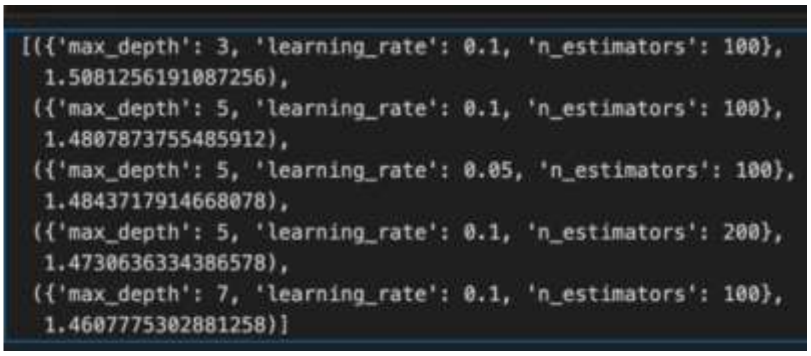
<u>Main iteration:</u>

1. <u>Data Preprocessing:</u>
   - Extracted the **station name** column and exploded it into individual words.
   - Analyzed word frequencies and distributions to identify **the top popular words** and from them we chose the words with top correlation to the label.
   - Created binary features indicating the presence of these popular words.
   - Encoded categorical variables (cluster, alternative, part, station_id, and trip_id_unique) using label encoding.
   - We also used the preprocessing we used on the baseline data.
   - Removed records where the target variable passengers_up exceeded 30 only from the train.
2. <u>Data Splitting:</u>

- For the main iteration we used more 15% of the total data.

3. <u>Modeling</u>
   - **Hyperparameter Tuning:**
     MAX_DEPTH = 7, LEARNING_RATE = 0.1, N_ESTIMATORS = 100
   - **The XGBoost model** with tuned hyperparameters outperformed other models, achieving the lowest MSE on the test set. XGBoost was selected for its robustness and ability to handle complex interactions in the data. The other models served as baselines and provided valuable insights into feature importance.

Loss of the XGBoost model:
As we can see – we iterate over some Hyperparameters and choose the ones that supply the minimize loss.



```
[({'max_depth': 3, 'learning_rate': 0.1, 'n_estimators': 100},
  1.5081256191087256),
 ({'max_depth': 5, 'learning_rate': 0.1, 'n_estimators': 100},
  1.4807873755485912),
 ({'max_depth': 5, 'learning_rate': 0.05, 'n_estimators': 100},
  1.4843717914668078),
 ({'max_depth': 5, 'learning_rate': 0.1, 'n_estimators': 200},
  1.4730636334386578),
 ({'max_depth': 7, 'learning_rate': 0.1, 'n_estimators': 100},
  1.4607775302881258)]
```

<u>Mission Two</u>

<u>Libraries Used:</u>
os, logging, argparse, pandas, xgboost, numpy, matplotlib, sklearn, geopy.

<u>Baseline</u>

1. **Data Preprocessing:**

We convert the data to be in trip_id unique resolution instead of station resolution. In addition we generate the label as the delta between the min arrive time and max arrive time per trip_id unique.

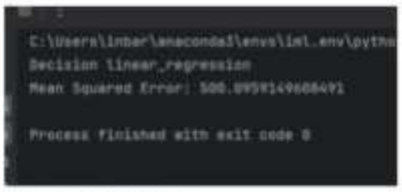
- Group by trip_id_unique and calculate statistics (e.g., station_cnt, total_passenger, mean_passenger, mean_passenger_c, start_time).
- Merge with other features (cluster, direction, mekadem_nipuach_luz) that were in trip_id unique level.
- Encode categorical variables.

2. **Data Splitting:**
   - For the baseline we used 10% of the data according to trip_id_unique groups.
3. **Modeling:**

we tried to use 3 different modeling:

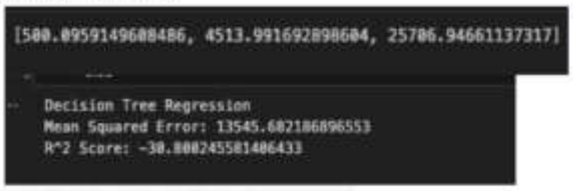
1. Decision Trees
2. Linear Regression
3. Polynomial Regression – We utilized a third-degree polynomial regression model.

Eventually, we selected Linear Regression due to its lowest mean squared error.



```
C:\Users\inbar\anaconda3\envs\iml.env\pytho
Decision linear_regression
Mean Squared Error: 500.0959149608491

Process finished with exit code 0
```

polynomial fitting



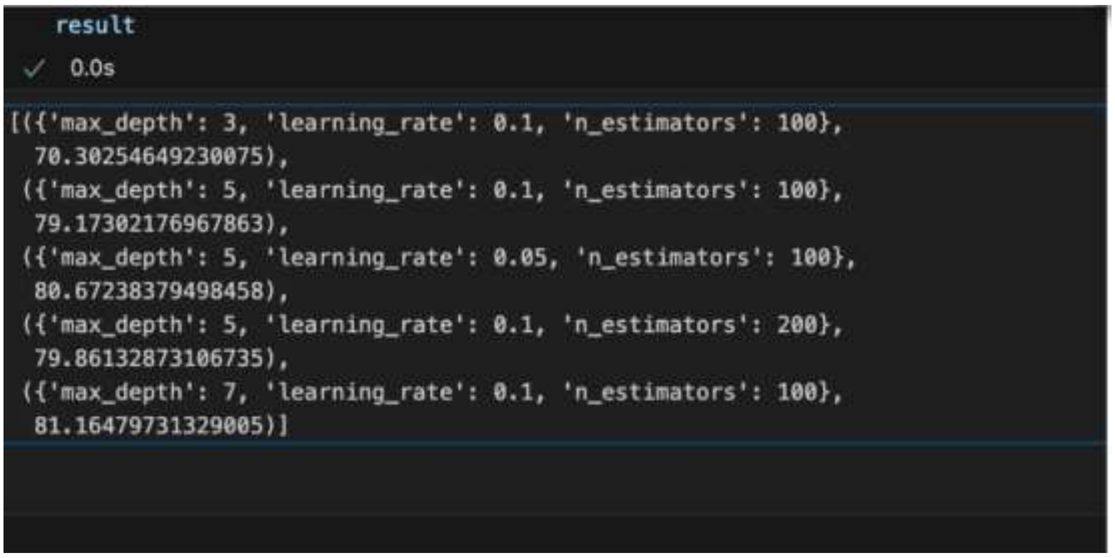
```
[500.0959149608486, 4513.991692898604, 25706.94661137317]

Decision Tree Regression
Mean Squared Error: 13545.682186896553
R^2 Score: -30.800245581406433
```

Main iteration:

1. Data Preprocessing:
   - Analyzed word frequencies and distributions to identify **the top popular words** and from them we chose the words with top correlation to the label- each such word discount as a feature.
   - Calculate using "geopy" the trip length(over the stations long and lat).
2. Data Splitting:
   - For this iteration we used 28% of the data according to trip_id_unique groups.
3. Modeling:
   - Hyperparameter Tuning:
     BASE_LINE_SAMPLE_SIZE = 0.05 , MAX_DEPTH = 3 , LEARNING_RATE = 0.1
     N_ESTIMATORS = 100.
   - **The XGBoost model** with tuned hyperparameters outperformed other models, achieving the lowest MSE on the test set. XGBoost was selected according to the same reasons we explained in the previous mission.

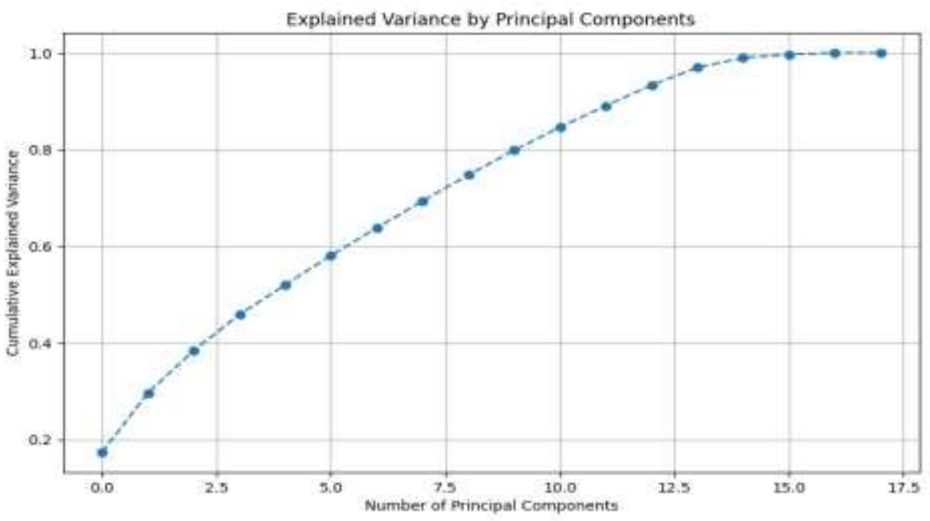
Loss of the XGBoost model:

As we can see – we iterate over some Hyperparameter and choose the ones that supply the minimize loss.

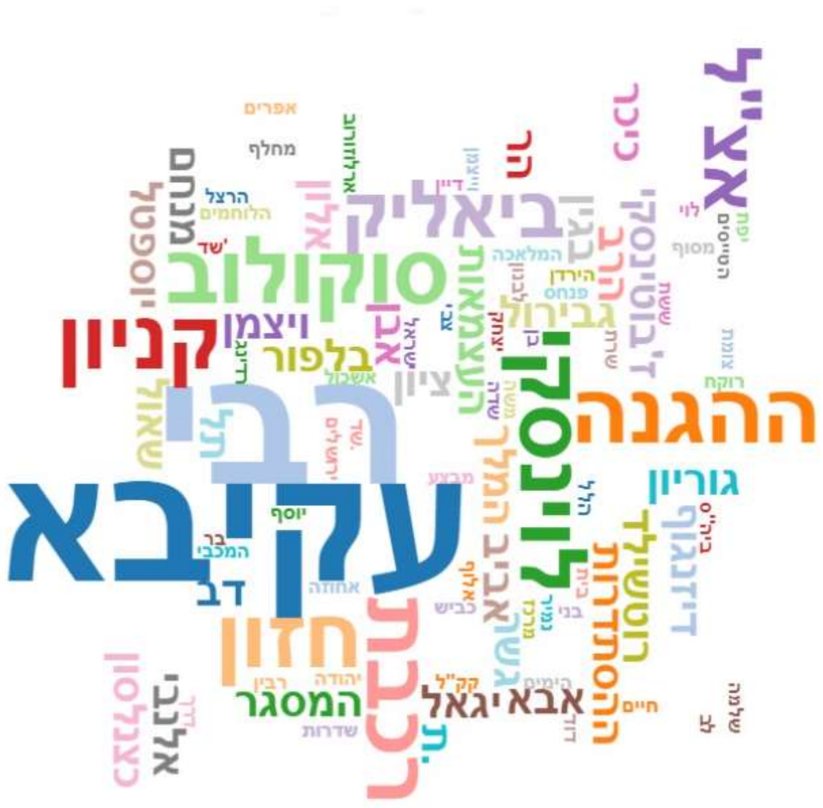


```
result
✓ 0.0s
[({'max_depth': 3, 'learning_rate': 0.1, 'n_estimators': 100},
  70.30254649230075),
 ({'max_depth': 5, 'learning_rate': 0.1, 'n_estimators': 100},
  79.17302176967863),
 ({'max_depth': 5, 'learning_rate': 0.05, 'n_estimators': 100},
  80.67238379498458),
 ({'max_depth': 5, 'learning_rate': 0.1, 'n_estimators': 200},
  79.86132873106735),
 ({'max_depth': 7, 'learning_rate': 0.1, 'n_estimators': 100},
  81.16479731329005)]
```

We used PCA to reduce the number of dimensions while preserving most of the variance in the data. This helps in simplifying the model and potentially improving its performance. We saw that we have 17 features and according to this plot we can see that we can use 15 features from the original features. We tried to explain which of the features are more weights according to label but we didn't have enough time.



In this plot we can see the most popular words, their size was determined by the passenger up median value.


עקיבא
רבי
קניון
סוקולוב
ביאליק
ההגנה
ז'בוטינסקי
חזון
המסגר
אבא
יגאל
ויצמן
בלפור
גוריון
גבירול
ציון
יוסף
דב
המכבי
אחוזה
אפרים
מחלף
הרצל
הלוחמים
המלאכה
הירדן
פנחס
לוי
מסוף
רוקח
אשכול
יהודה
רבין
שדרות
קק"ל
הימים
כביש
חיים
בני
בר
מבצע