1. Data pre-processing
   - **OneHot encoding**
   - **Data normalization**
   - Dimension reduction (use dimension reduction results to compare accuracy, such as F1 score for classification and MSE for regression)
     - Correlation coefficients, such as heatmap
     - PCA
     - Lasso
     - RFE: recursive feature elimination
   - EDA

2. Machine Learning Models - Supervised learning
80%-20% train-test splitting
Actually, I first randomly split a test set with 999 samples and held it back from the model training process. For the rest 10,000 samples, I did 80%-20% splitting for training and validation. (from Yiqun)

Cross-validation will only be used during the training process. And we use test accuracy to compare model performance.
   - Neural Networks for small dataset – Noushin
     - Classification on whether the delivery will reach on time: predict "Reached.on.Time_Y.N"
     - Regression on customer ratings: "Customer_rating"
   - Logistic regression (similar to 1-layer NN) - classification on "Reached.on.Time_Y.N" - Noushin
   - Decision Tree Classifier - classification on "Reached.on.Time_Y.N" - Jinqin
     - Features: all columns except "ID" and "Reached.on.Time_Y.N"
     - Target variable: "Reached.on.Time_Y.N"
   - Decision Tree Regressor - regression on "Customer_rating" - Yiqun
   - Bagging trees and Random forest - Yiqun
     XGBoost
   - KNN - classification on "Reached.on.Time_Y.N" - Jinqin - Lecture 4 / search on Google "KNN tutorial"

3. Machine learning models - unsupervised learning
   - K-means

4. Evaluation
   - Regression models: MAE
   - Classification models: confusion matrix and calculate f1 scores
   - Compare model performance: plot accuracy wrt models - bar chart
   - ~~80%-20% train-test splitting.~~ **A test set with 999 samples to evaluate model performance. (from Yiqun)**
   - Cross-validation will only be used during the training process. And we use test accuracy to compare model performance.

   - A technique that transform regression error to confusion matrix

~~y= 1,2,3,4,5~~
~~y^ = 1.1, 2,8, 3, 3,8, 7~~
~~err = 0.1, 0.8, 0, -0.2, 2~~
~~Err2 = 0, 1, 0, 0, 2~~
~~Sign = +, +, 0, -, +~~
~~TP, FP, TP/TN, TN, FP~~

Result of Exploratory Data Analysis (EDA)

*** The original image files for all figures shown below have been uploaded to the Google Drive folder "CS539_project".*

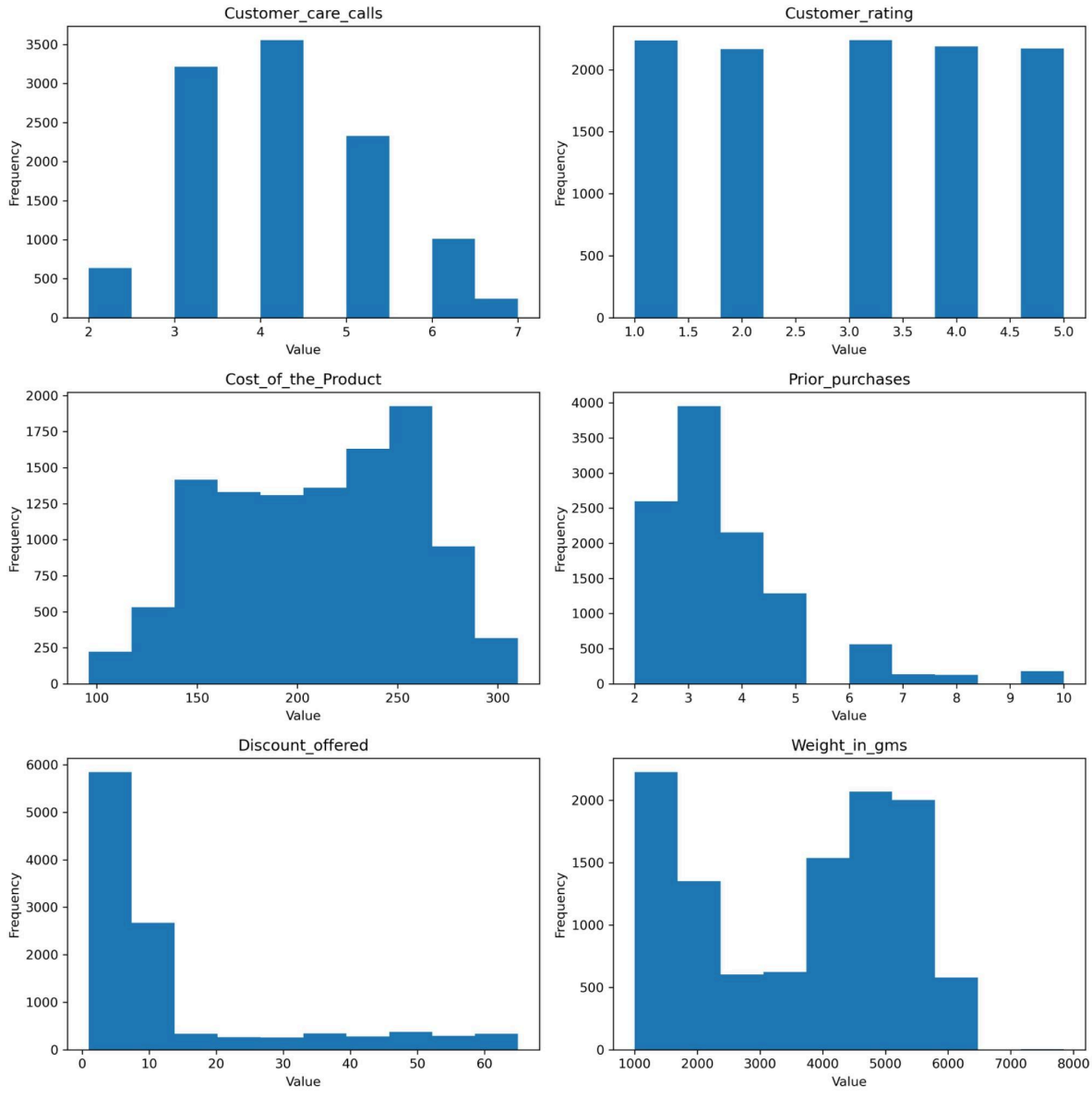


Figure 1. Histograms of six numerical features

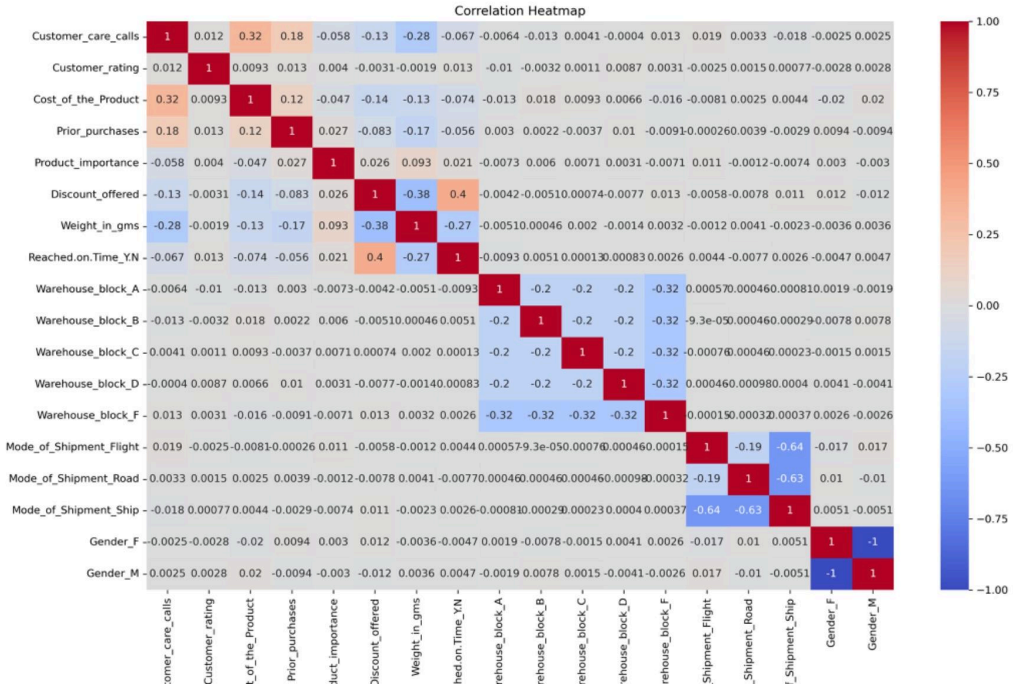


Figure 2: Heatmap displaying the relationships among all variables. The visualization indicates the absence of linear correlations between our target variables and the predictors. Each cell represents the correlation coefficient between pairs of variables, with warmer colors denoting positive correlation values and cooler colors indicating negative correlations. This suggests that the relationship between the target variables and predictors is not characterized by linear trends, emphasizing the complexity of the underlying data structure.

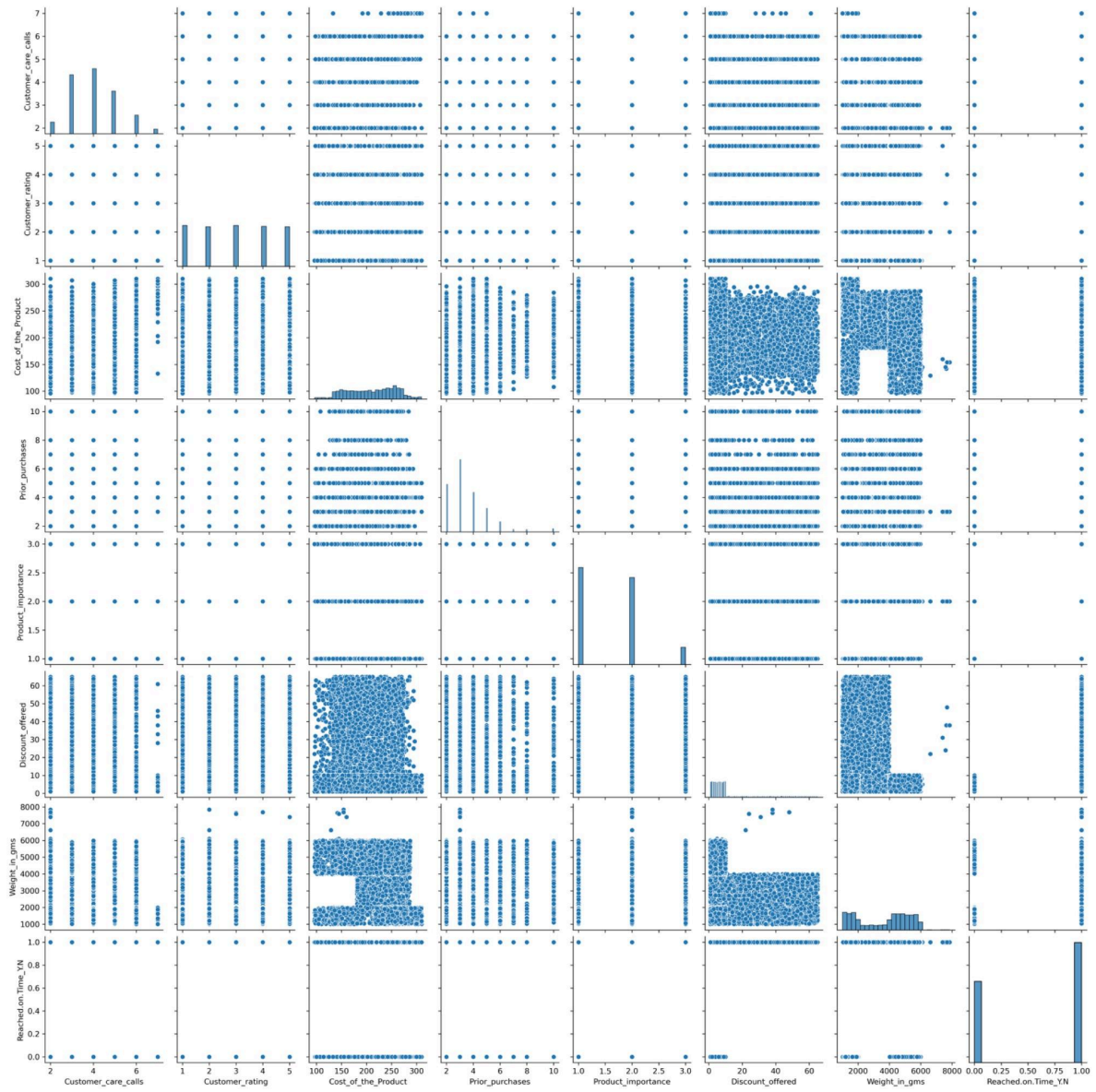


Figure 3: Pairplot illustrating the distribution of individual variables and their pairwise relationships. Each subplot provides a visual representation of the distribution of a single variable, while the scatterplots showcase the interactions between different variables.

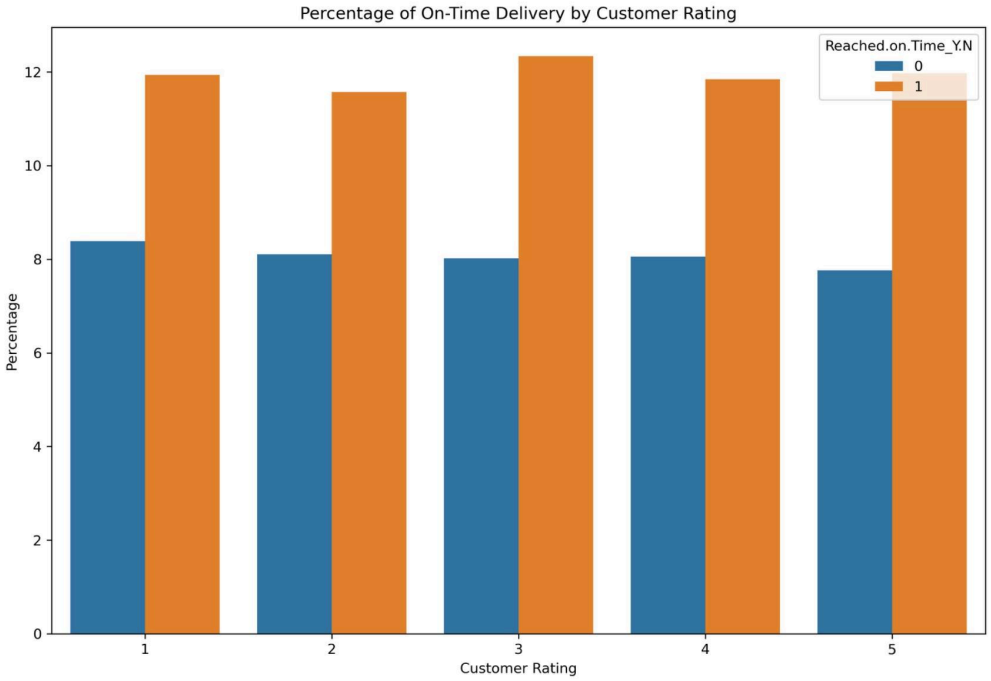


Figure 4: Percentage countplot depicting on-time delivery rates categorized by customer rating. Each bar represents the proportion of on-time deliveries within each customer rating category, offering insights into the relationship between customer satisfaction levels and delivery punctuality.

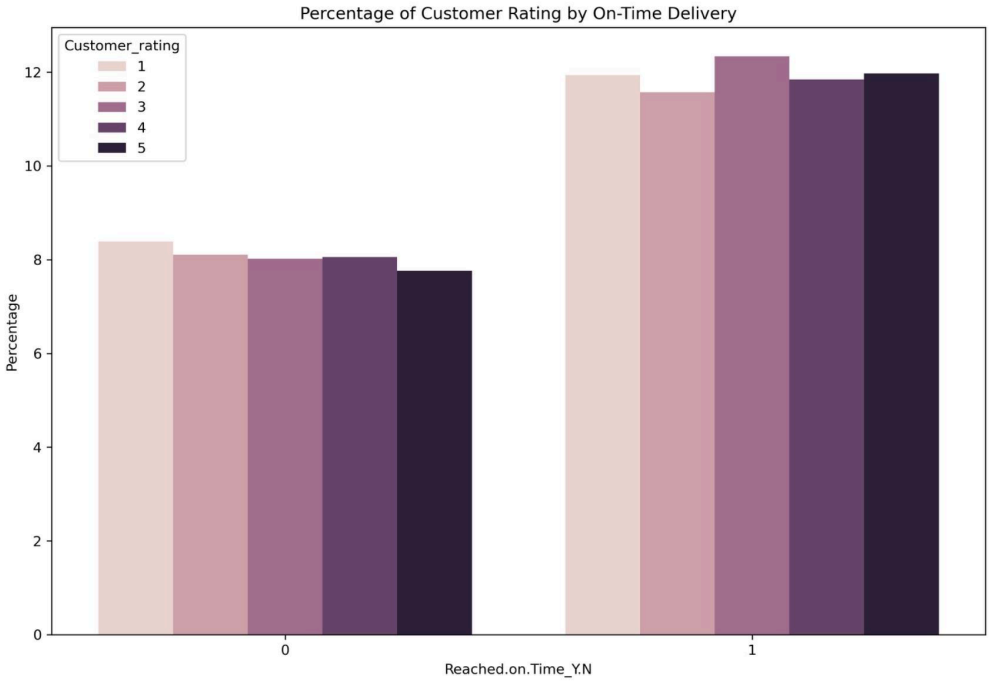


Figure 5: Percentage countplot depicting customer rating categorized by on-time delivery rates. Each bar represents the proportion of one to five customer ratings within the label of on-time deliveries, offering insights into the relationship between customer satisfaction levels and delivery punctuality.

Result of Decision Tree Classification

Metrics	Before Optimization	After Optimization(GridSearchCV)
Accuracy	0.6559	0.6645
F1-Score	0.68	0.71
Best Parameters	N/A	{max_depth: 10, max_features: None, min_samples_leaf: 10, min_samples_split: 2}

Confusion Matrix

Actual \ Predicted	no	yes
no	691	204
yes	534	771

Result of KNN

Metrics	Accuracy	F1-Score
Result	0.63	0.68

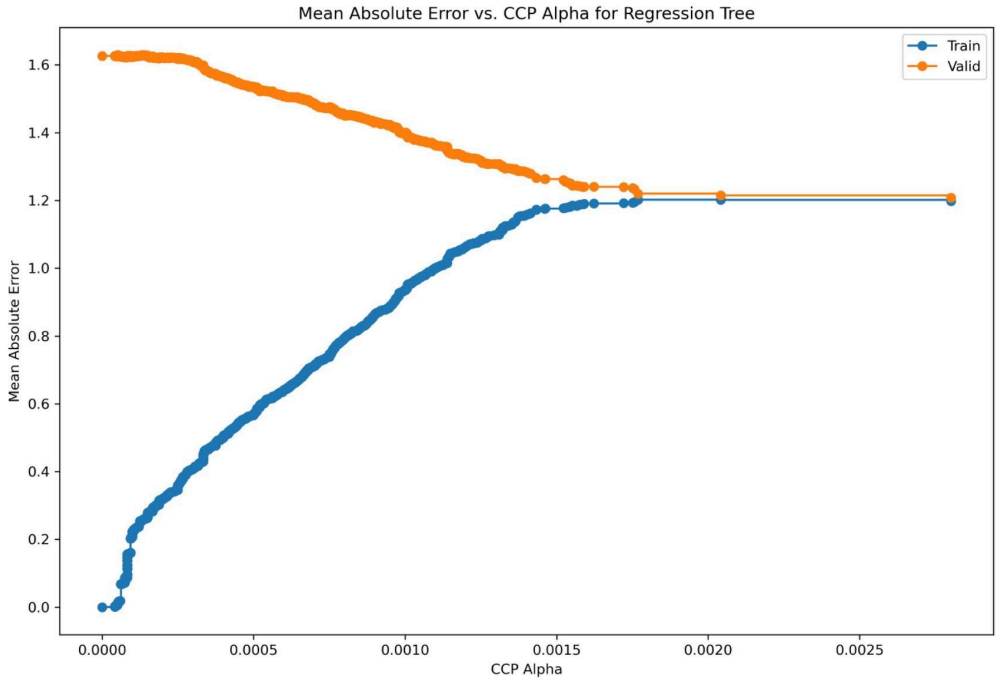
Confusion Matrix

Actual \ Predicted	**Positive**	**Negative**
Positive	524	371
Negative	436	869

two model’s performance:

model	accuracy	F1_score
Classification decision tree	0.66	0.68
KNN	0.63	0.68

Result of Decision Tree Regressor for Customer Rating Prediction (Regression Task)



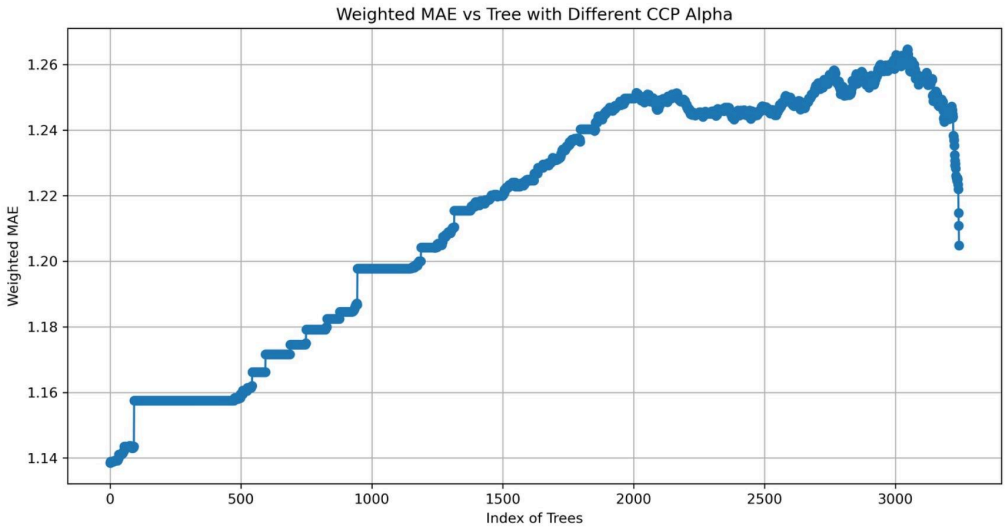
In [174]:
```
text_representation = export_text(best_tree)
print(text_representation)
```

```
|--- value: [3.01]
```

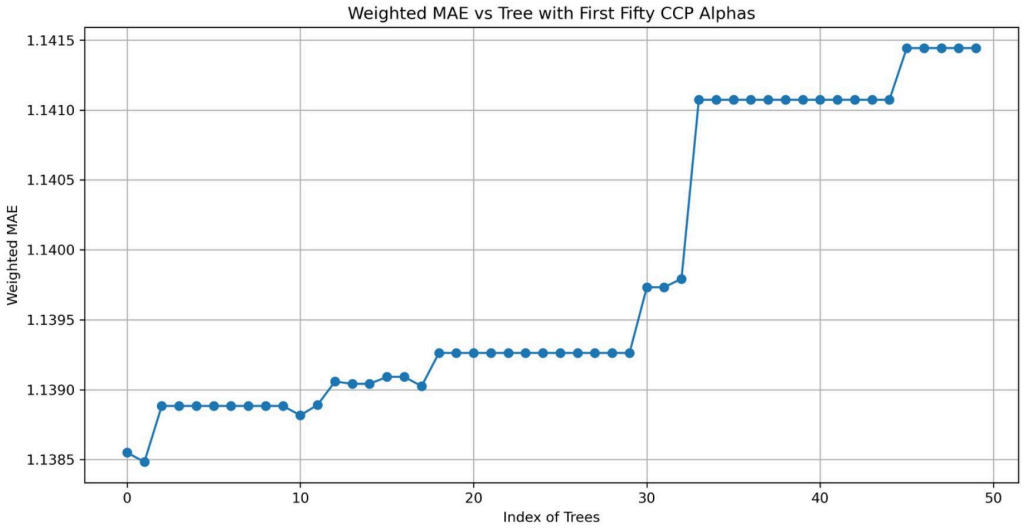
This result shows that the decision tree with lowest validation error predicts customer rating as a fixed value, 3.01, regardless of the input predictors. That is, it is underfitting the data, failing to capture the complexities of the relationship between predictors and the target variable.
In order to fix this problem, we defined a weighted error to make training loss involved. The weighted error of a decision tree regressor is defined as 30% of training loss plus 70% of the validation error.

In [209]:
```
err_combined = []
for x, y in zip(train_errors, valid_errors):
    err_combined.append(x*0.3 + y*0.7)
```

And next, we plot this weighted MAE for each decision tree regressor.



And we found the one with lowest weighted error, which is the second tree in this 1-50 index window. We called it the “balanced tree”.



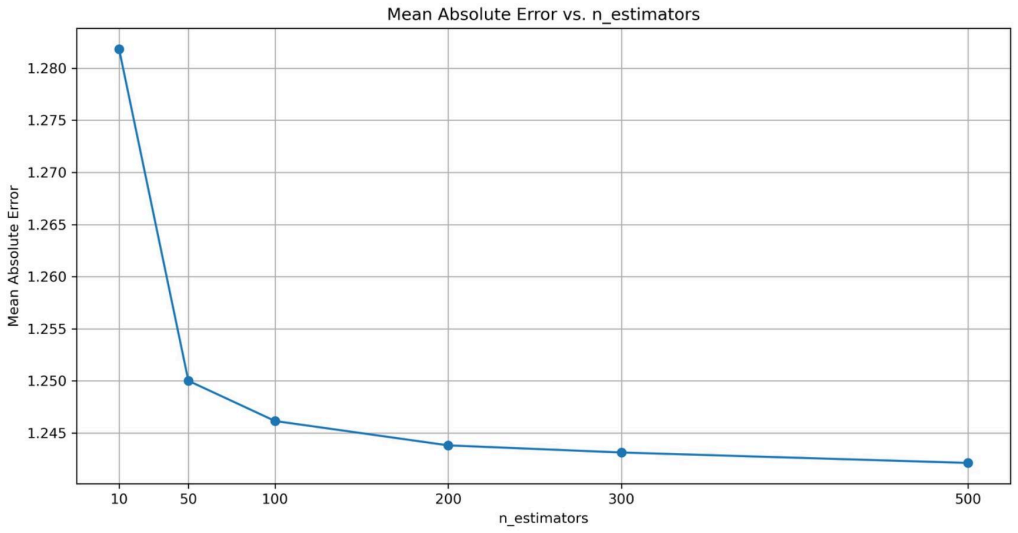

```
In [222]:
1 y_test_pred = balanced_tree.predict(X_test)
2 balanced_tree_test_err = mean_absolute_error(y_test, y_test_pred)
3 balanced_tree_test_err
```

```
Out[222]: 1.5945945945945945
```

```
Feature importance:
Feature 0 - Customer_care_calls - 0.06944723883444412
Feature 1 - Cost_of_the_Product - 0.20332870053411448
Feature 2 - Prior_purchases - 0.04838938350795061
Feature 3 - Product_importance - 0.041916733735356204
Feature 4 - Discount_offered - 0.1431705590323048
Feature 5 - Weight_in_gms - 0.29382669485445523
Feature 6 - Reached.on.Time_Y.N - 0.021482793053758473
Feature 7 - Warehouse_block_A - 0.023663663321023952
Feature 8 - Warehouse_block_B - 0.01916654150249963
Feature 9 - Warehouse_block_C - 0.02094258869612339
Feature 10 - Warehouse_block_D - 0.014130412985467154
Feature 11 - Warehouse_block_F - 0.02436370367142376
Feature 12 - Mode_of_Shipment_Flight - 0.017156898711657893
Feature 13 - Mode_of_Shipment_Road - 0.016618976312284577
Feature 14 - Mode_of_Shipment_Ship - 0.016140714008025112
Feature 15 - Gender_F - 0.012240954702465338
Feature 16 - Gender_M - 0.01401344253664535
```

Result of Random Forest Regressor for Customer Rating Prediction (Regression Task)

The plot below shows the validation MAE of a random forest regressor with respect to different values of n_estimators.

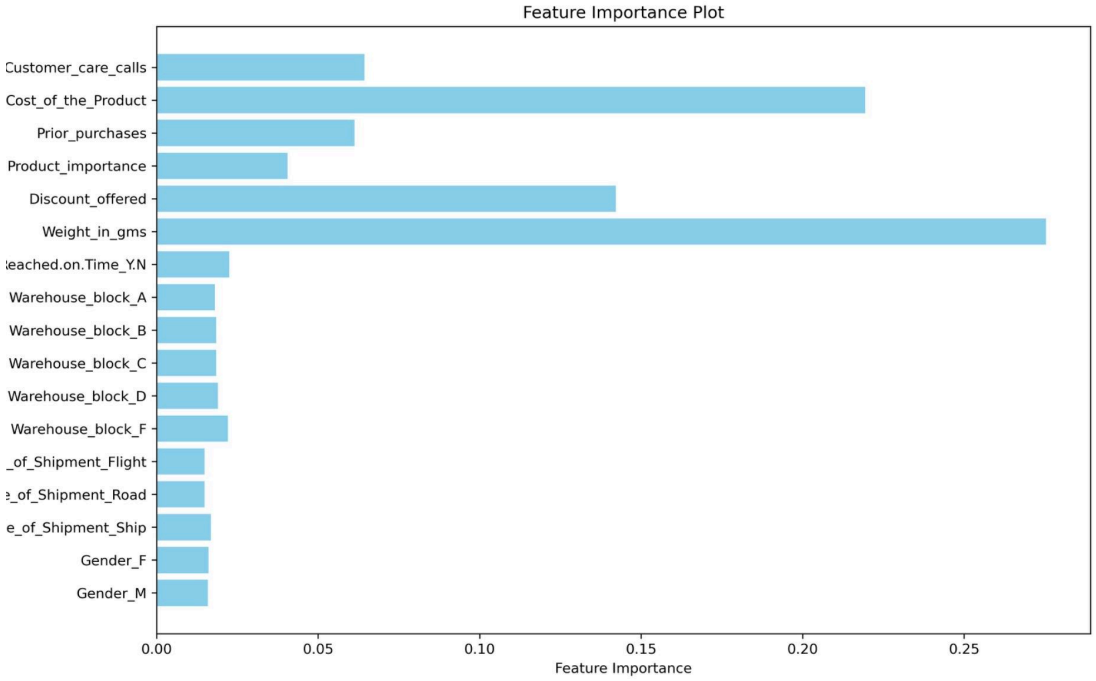


Best random forest regressor is the one with 500 n_estimators. And the test MAE of this best model is 1.217.

```
In [235]:
1 y_pred_rfr = best_rf_regressor.predict(X_test)
2 best_rf_test_err = mean_absolute_error(y_test, y_pred_rfr)
3 best_rf_test_err
```

```
Out[235]: 1.217169169169169
```

Feature importance of the best random forest regressor:

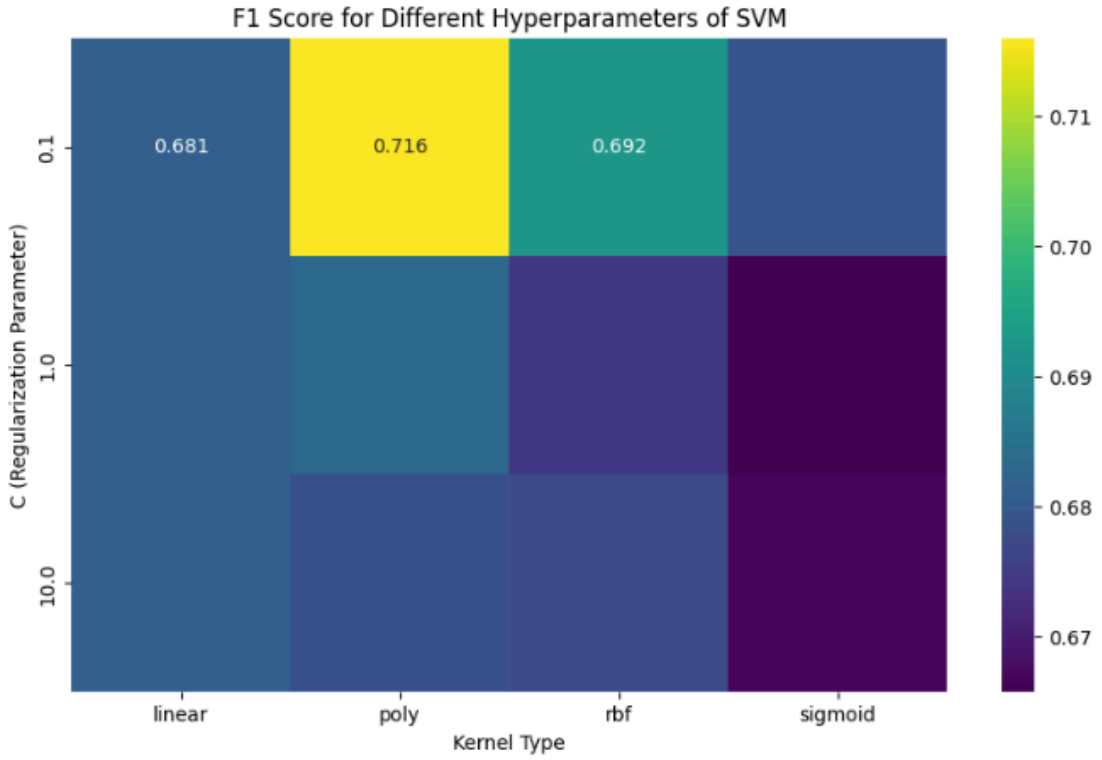


1. Jinqin
    a. Use “delivery_readable.csv” to train decision tree classifier ☑️
    b. Normalize data → train KNN ☑️
    c. Confusion matrix and f1 score ☑️

	Decision Tree Classifier	KNN	SVM	Neural Network (Multi-Layer perceptron)	Logistic Regression
Confusion matrix	[[691 204] [534 771]]	[[524 371] [436 869]]	[[493 402] [370 935]]	[[564 331] [477 828]]	[[539 356] [409 896]]
F1 score	0.71	0.68	0.71	0.67	0.70

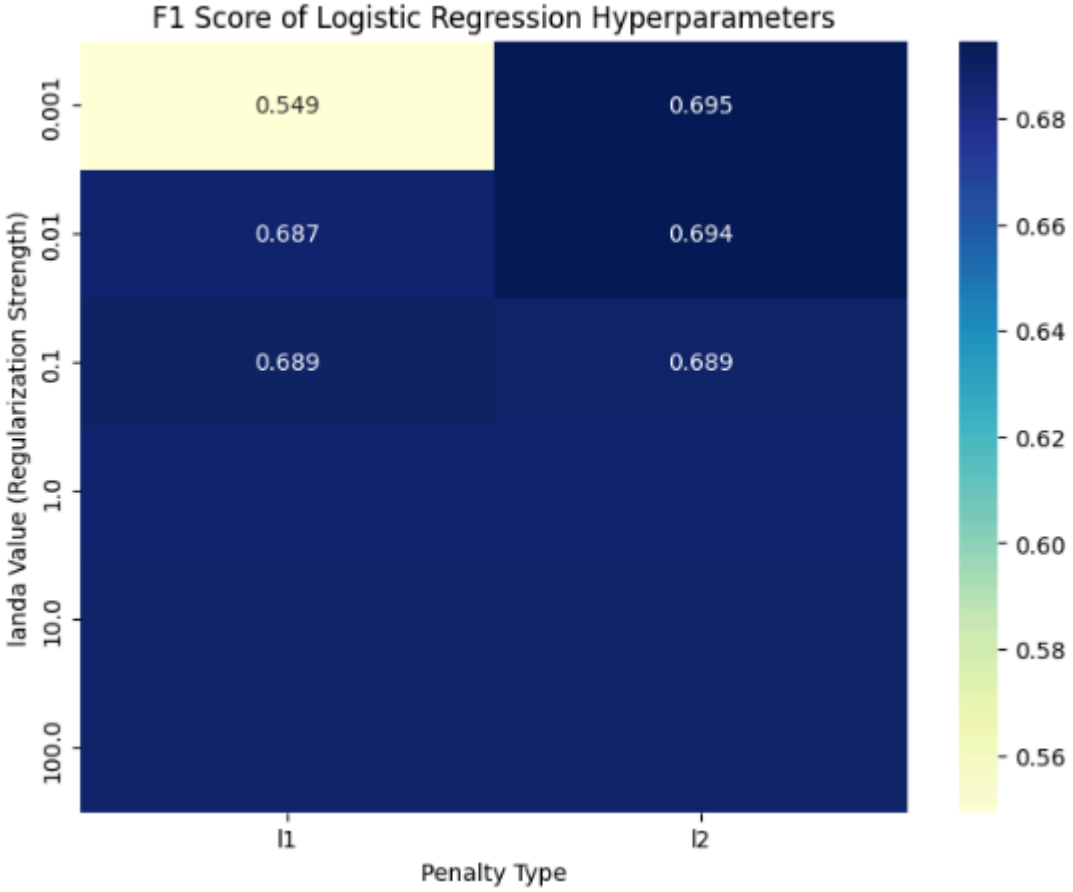
SVM (classification):

Heatmap: Performance of different combinations of kernel types and regularization parameters C



NN (multilayer perceptron) for classification:
Similar to logistic regression (having 2 layers) but tuning the density, using dropout technique, tuning ADAM and SGD optimizer, Results show overitting

Logistic regression (classification):



Neural Networks for small dataset – Noushin:
Problem: Issue of overfitting
Solutions:

- Good idea: Transfer learning (but there is only one dataset available)
- Using small NN with fewer number of Neurons and layers

- Utilizing regularization techniques
- Using dropout technique

Weaker performance of NN with two layers compared to logistic regression in terms of f1 score. Graphs also show overfitting issue.

2. Noushin
    a. SVM for classification
    b. Confusion matrix and f1 score
    c. Share ideas and thoughts about challenges, future work, what we learned and discussed from papers (NN limitation on small dataset)
    d. Hyperparameter tuning plot

3. Yiqun
    a. Naive Bayes (I gave up)
    b. Compare regression result before and after normalization
    c. Compare model performance using MAE/MAPE
    d. aIdeas and thoughts about tuning decision tree regressor

	Decision Tree Regressor	Random Forest Regressor	Neural Network
Test MAE	1.594	1.217	1.219

4. Ronit
    a. Build up project website
    b. In our presentation slides, finish pages of “background”, “objectives”, and “project website”.