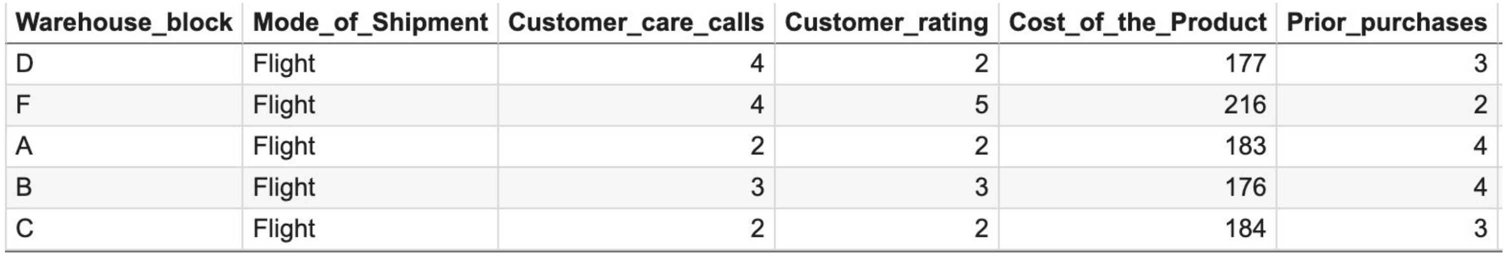
Can We Predict if a Shipment Will be On Time?

ISyE 604

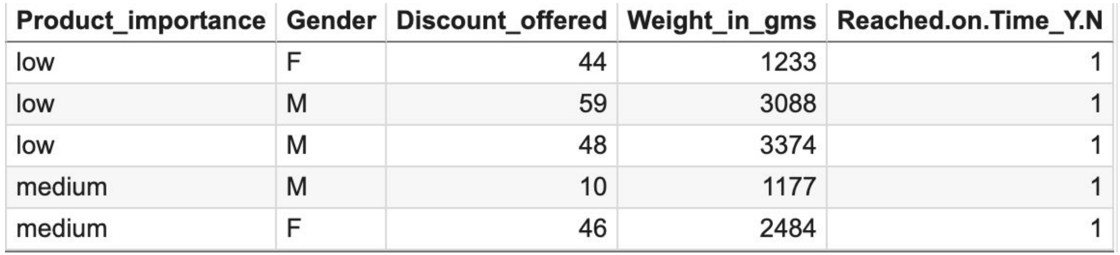
Praharshith Jamalapuram, Aaditya Padmanabhan, Kaviya Ramanathan, Sinfeney Teng

Introduction to the Data

- Customer database from an e-commerce company
- 10999 entries, 11 features



Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases
D	Flight	4	2	177	3
F	Flight	4	5	216	2
A	Flight	2	2	183	4
B	Flight	3	3	176	4
C	Flight	2	2	184	3



Product_importance	Gender	Discount_offered	Weight_in_gms	Reached.on.Time_Y.N
low	F	44	1233	1
low	M	59	3088	1
low	M	48	3374	1
medium	M	10	1177	1
medium	F	46	2484	1

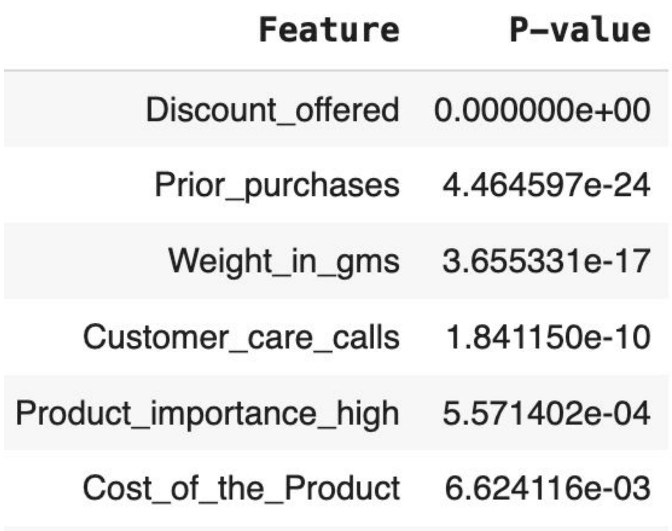
Feature Selection

Features(X):

- **Discount_Offered**: Discount offered on that specific product
- **Customer_care_calls**: The number of calls made for enquiry of the shipment
- **Prior_purchases**: The Number of Prior Purchase
- **Weight_in_gms**
- **Product_importance_high**
- **Cost_of_the_Product**

Target(Y):

- **Reached_on_time_N_Y**

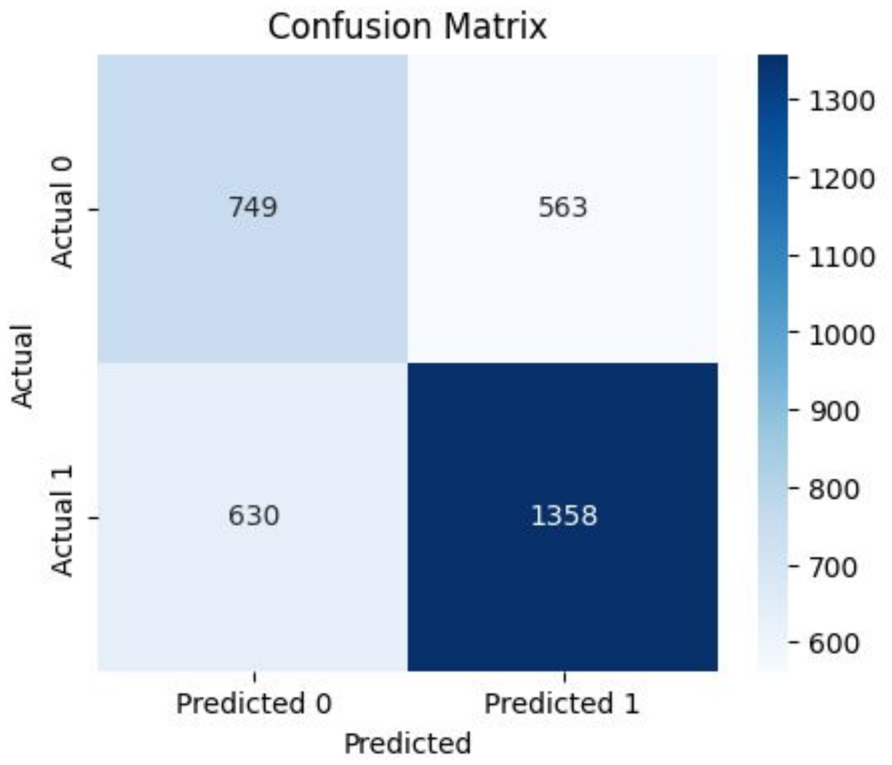


Feature	P-value
Discount_offered	0.000000e+00
Prior_purchases	4.464597e-24
Weight_in_gms	3.655331e-17
Customer_care_calls	1.841150e-10
Product_importance_high	5.571402e-04
Cost_of_the_Product	6.624116e-03

Reasons for Picking the Methods

- Logistic Regression
  - Target Y variable is a prediction of binary outcome, which makes LR suitable
  - Can handle overfitting using penalty='L1' (Lasso) or penalty='L2' (Ridge)
- Decision Tree
  - Larger sample size
  - Works well with low-dimensional data

Logistic Regression



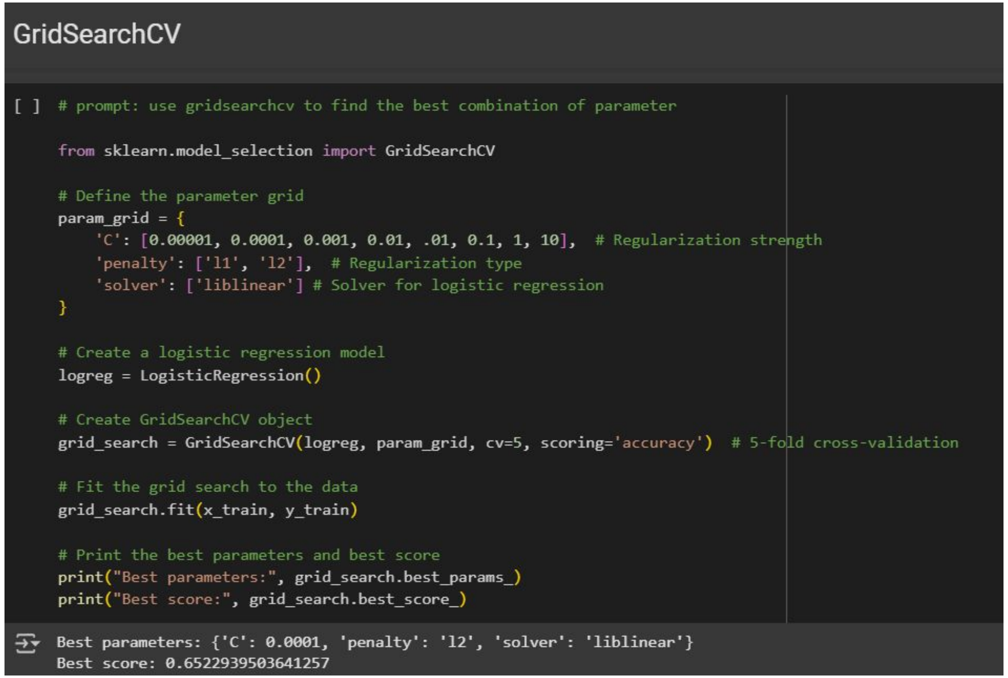
Precision for class 0 is 0.54, meaning 54% of predictions for class 0 were correct.

Recall for class 0 is 0.58, indicating that 58 % of actual class 0 instances were correctly identified.

The F1-score balances precision and recall, providing an overall measure of model performance.

The accuracy of the model is 64%

Logistic Regression

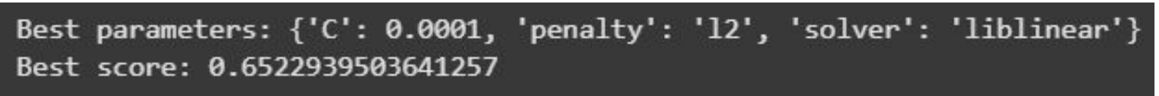


GridSearchCV

```
# prompt: use gridsearchcv to find the best combination of parameter

from sklearn.model_selection import GridSearchCV

# Define the parameter grid
param_grid = {
    'C': [0.00001, 0.0001, 0.001, 0.01, .01, 0.1, 1, 10],  # Regularization strength
    'penalty': ['l1', 'l2'],  # Regularization type
    'solver': ['liblinear'] # Solver for logistic regression
}

# Create a logistic regression model
logreg = LogisticRegression()

# Create GridSearchCV object
grid_search = GridSearchCV(logreg, param_grid, cv=5, scoring='accuracy')  # 5-fold cross-validation

# Fit the grid search to the data
grid_search.fit(x_train, y_train)

# Print the best parameters and best score
print("Best parameters:", grid_search.best_params_)
print("Best score:", grid_search.best_score_)
```

```
Best parameters: {'C': 0.0001, 'penalty': 'l2', 'solver': 'liblinear'}
Best score: 0.6522939503641257
```

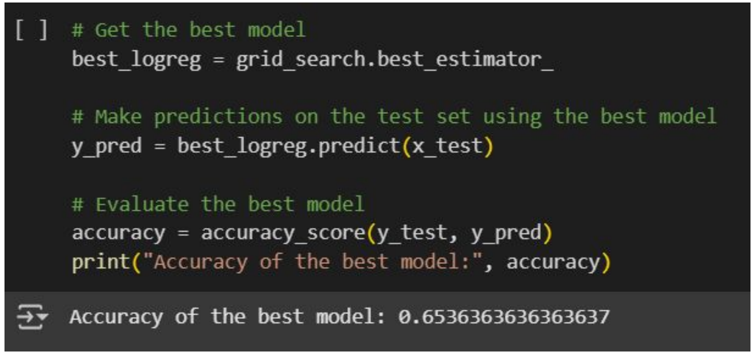
The selection of the best parameter using the cross-validation selection (GridSearchCV).



```
Best parameters: {'C': 0.0001, 'penalty': 'l2', 'solver': 'liblinear'}
Best score: 0.6522939503641257
```

Now, we train and predict the accuracy once again with the best parameters

Logistic Regression

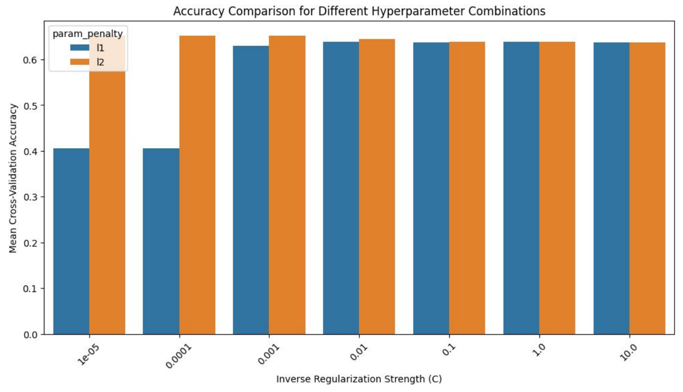


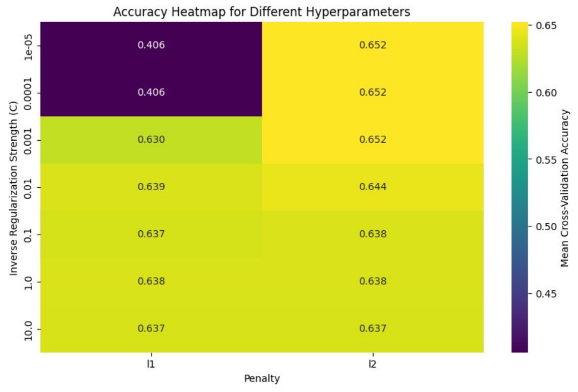
```
[ ] # Get the best model
    best_logreg = grid_search.best_estimator_

    # Make predictions on the test set using the best model
    y_pred = best_logreg.predict(x_test)

    # Evaluate the best model
    accuracy = accuracy_score(y_test, y_pred)
    print("Accuracy of the best model:", accuracy)

    Accuracy of the best model: 0.6536363636363637
```

HyperParameters:
C (Inverse Regularization Strength): [0.00001, **0.0001**, 0.001, 0.01, 0.1, 1, 10]
Penalty (Regularization Type): ['l1', '**l2**']
Solver:['**liblinear**']

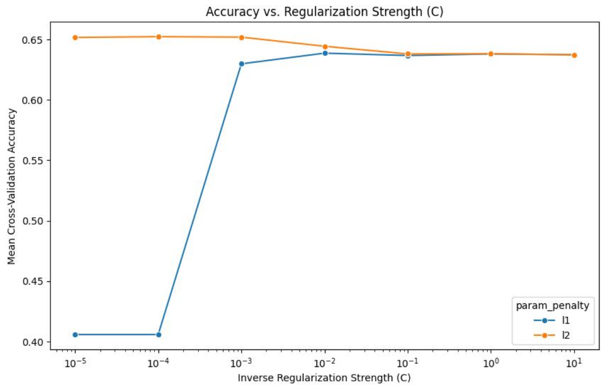
To understand this and visualize the comparison of tuning hyperparameters we have the following:

1. A bar graph (L1 and L2)
2. Heatmap and (L1 and L2)
3. Line Graph (Inverse Reg. Strength)





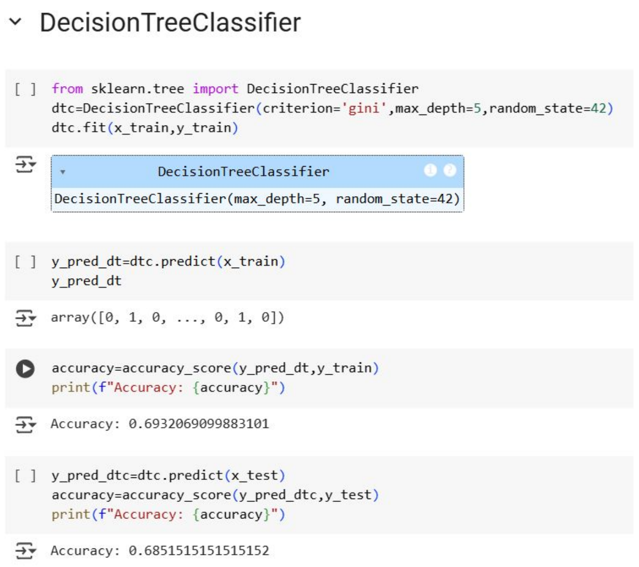
Inverse Regularization Strength (C)	l1	l2
1e-05	0.406	0.652
0.0001	0.406	0.652
0.001	0.630	0.652
0.01	0.639	0.644
0.1	0.637	0.638
1.0	0.638	0.638
10.0	0.637	0.637



Findings and Analysis: Visual Evidence

- **L2 Regularization consistently provides better and stable performance** across all tested values of C.
- **L1 Regularization is sensitive** to the choice of C — it underperforms with strong regularization (small C) but improves with weaker regularization (larger C).
- **Optimal C values** are in the **range of 0.0001 to 0.01**, where both penalties achieve their highest accuracies — but **L2 still outperforms L1** slightly.
- **Over-regularization (very small C)** hurts L1 much more than L2

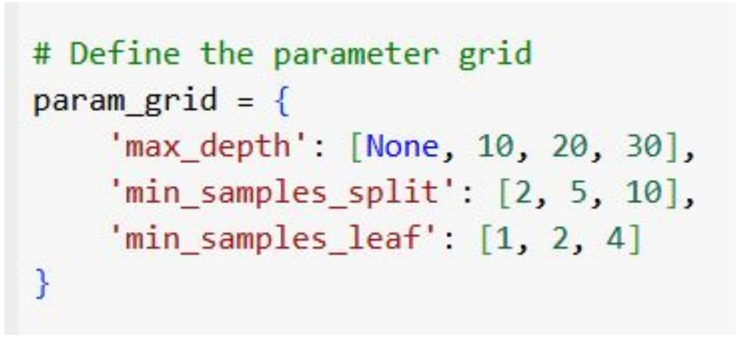
Decision Tree



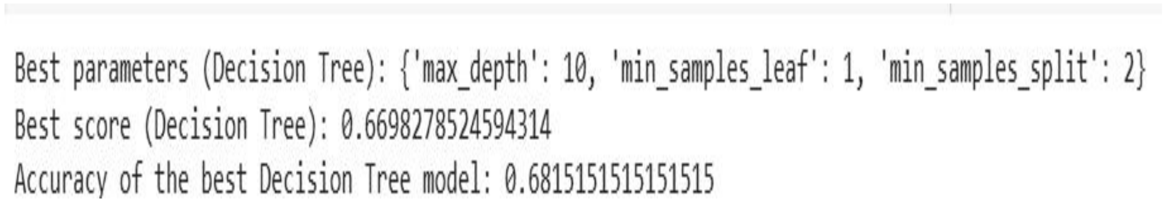
DecisionTreeClassifier

```
[ ] from sklearn.tree import DecisionTreeClassifier
    dtc=DecisionTreeClassifier(criterion='gini',max_depth=5,random_state=42)
    dtc.fit(x_train,y_train)

    DecisionTreeClassifier
    DecisionTreeClassifier(max_depth=5, random_state=42)

[ ] y_pred_dt=dtc.predict(x_train)
    y_pred_dt

    array([0, 1, 0, ..., 0, 1, 0])

    accuracy=accuracy_score(y_pred_dt,y_train)
    print(f"Accuracy: {accuracy}")

    Accuracy: 0.6932069099883101

[ ] y_pred_dtc=dtc.predict(x_test)
    accuracy=accuracy_score(y_pred_dtc,y_test)
    print(f"Accuracy: {accuracy}")

    Accuracy: 0.6851515151515152
```

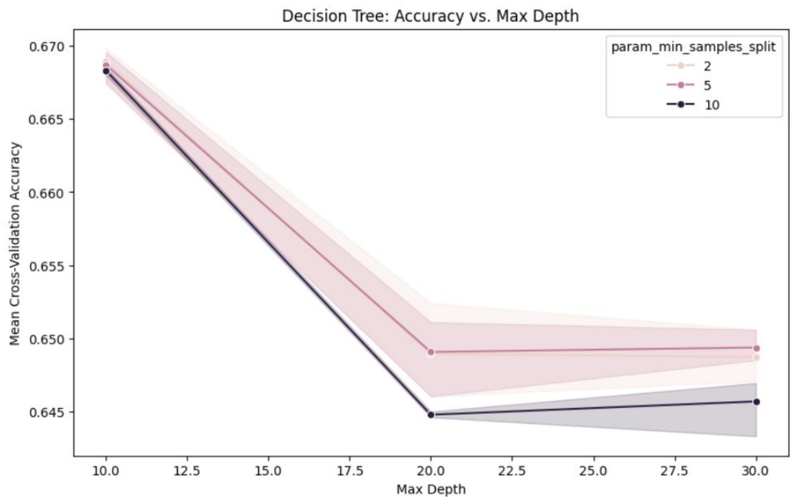
Grid Search CV:

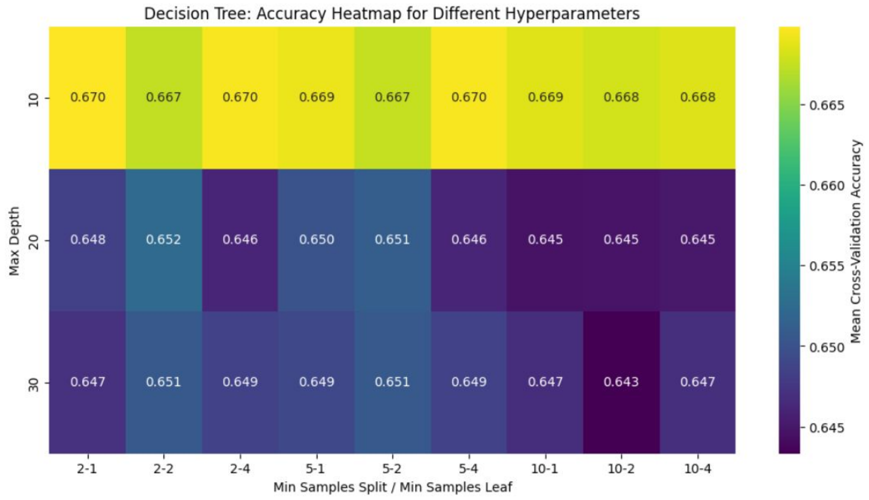


```
# Define the parameter grid
param_grid = {
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

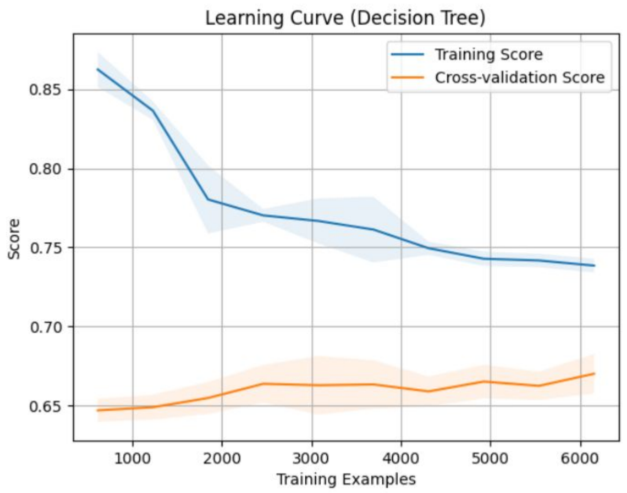


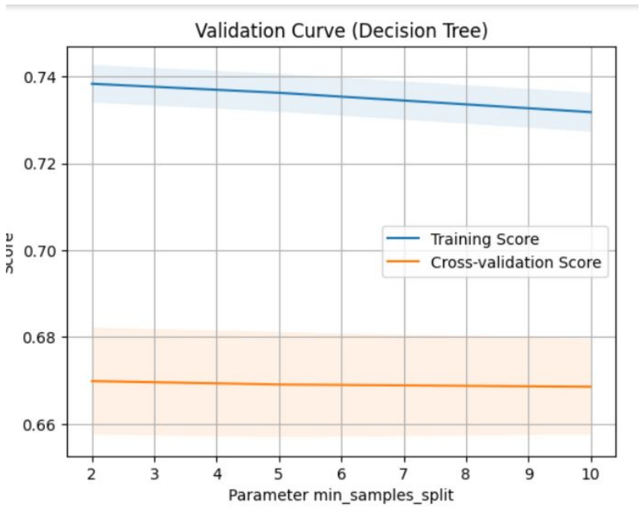
```
Best parameters (Decision Tree): {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2}
Best score (Decision Tree): 0.6698278524594314
Accuracy of the best Decision Tree model: 0.6815151515151515
```





- Hyperparameters: Max_depth = 10, min_samples_split = 2, min_samples_leaf =1
- Cross validation accuracy : 66.98%
- Test set accuracy with best model : 68.15%
- Higher min_samples_split values let to underfitting
- Overfitting observed beyond max_depth=10





Observations on Learning Curve

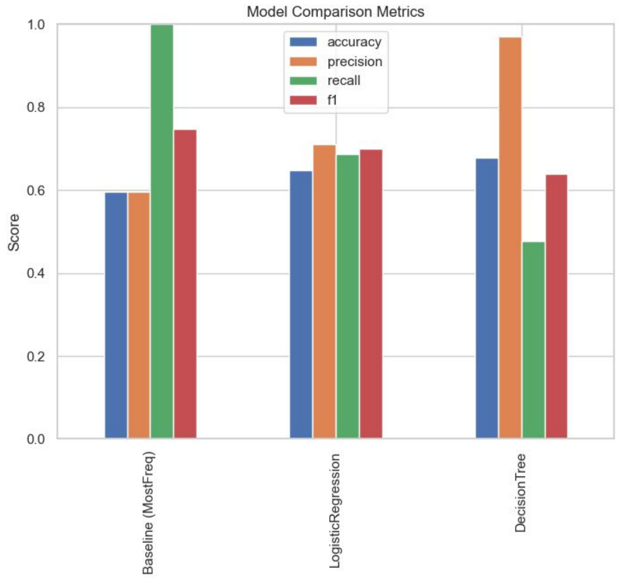
- Overfitting detected: high training score vs. low validation score.
- Cross-validation score improves slowly as training size increases
- There is an overfitting which is visible at the beginning but as the sample size increases the overfitting will reduce.

Observations on Validation Curve

- Lower min_samples_split improves training accuracy but risks overfitting.
- Higher values simplify the model but may slightly improve validation performance.
- Ideal balance around min_samples_split = 5 to reduce variance without hurting performance.

Conclusion

- Results:



- The Accuracy of LR Model: ~65%
- The Accuracy of DT Model: ~68%
- The Accuracy of Baseline (MostFreq) Model: ~60%

The Logistic Regression & Decision Tree models beat the baseline model on average of 5% difference. That translates to over 12.5% reduction in relative error. In real time, this would mean an overall improvement of at least ~5% in business aspects.

Conclusion

- *Can We Predict Whether a Shipment Will be On Time?* (Machine Learning Usability)
  - Yes. The Models do surpass the baseline by quite a margin, so they can be utilised. However, ~65-68% shows that there's a lot of room for improvement.
- *Why is the accuracy modestly at high 60s and not higher?* (Areas for improvement)
  - This could be attributed to the dataset properties & the models utilised in the project. While about 11,000 units of data is not small, there exists relatively weak correlations between the target variable and the other features of the data. Therefore, the models do not have enough to pull out the gaps between data.
- *What is the way forward?* (Future Possibilities)
  - Gathering more data with higher correlations with the target variable, such as weather info, traffic trends, carrier performance and so on.
  - ~~Using more complex ML models in order to discover deeper correlations within the data. Often times, complex models are rather not enthusiastically dealt with due to their increased training complexity and time consumed. However, in order to improve accuracy, both dataset and training models need to work hand in hand for better yielding results.~~
  - We originally thought that maybe with more complex models, we can gain higher accuracy. However, this is not all the case; sometimes more complex models can only lead to lower accuracy or remain the same.