Airline Customer Satisfaction Prediction

A **Bigger** Binary Classification Problem

Flight Crew

DATA ANALYSTS

- Dreama Wang - 261112206 (dreamaWang)
- Mathilda Zhang - 261112212 (lindasusany)
- Haoying Xu - 261109413 (HaoyingXu)
- Oscar Montemayor - 261082079 (oscarmse)

DATA SCIENTISTS

- Riley Zhu - 261094733 (RileyXiaoyu)

DATA ENGINEER

- Vibhu Bhardwaj - 261113187 (Bvibhu)
- Nishi Nishi - 261078870 (nishinishi06)

PRODUCT MANAGERS:

- ShanShan Lao - 261072808 (shanshanlao)
- Utkarsh Nagpal - 261071466 (Utkarshnagpal)

CUSTOMER SATISFACTION

1.4 Billion

USD Per Year

Revenue each US airline leaves on table by failing to improve their customer experience

Source: Forrester

NET PROMOTER SCORE PROGRAMS

200,000

USD Per Year

Amount spent on running NPS programs for a company with 1000 employees and $100 million in revneue

Source: CustomerGauge

Destination

Objective:

1) Predict Customer Satisfaction

2) Identify features that contribute most to customer satisfaction

Dataset:

1) Balanced Classification Dataset

2) Binary labels in target column named "satisfaction"

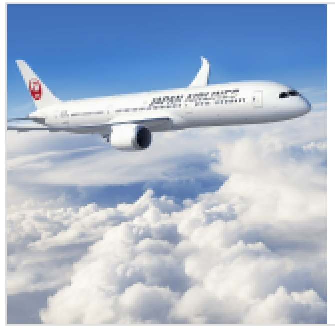
3) Source: Kaggle

Outcome:

1) Increased Revenue (upto USD 1.4B)

2) Cost savings in running NPS programs

3) Improved brand image and perception



Airline Passenger Satisfaction

What factors lead to customer satisfaction for an Airline?

k kaggle.com

https://www.kaggle.com/datasets/teejmahal20/airline-passenger-satisfaction?select=train.csv

Flight Path

Previous Path

01	02	03	04	05
EDA	**Preprocess**	**Supervised Learning**	**Semi-Supervised Learning**	**Overfitting Check**
Descriptive statistics, data exploration, correlation tests etc.	KNN impute, Outlier removal, encoding, test train validation split etc.	log reg, tree, boosting models on labelled dataset	log reg, tree, boosting models on unlabelled dataset	Learning Curve and validation curve for the TOP 2 model

New Path

06	07	08	09	10	11
Hyperparameter Tuning	**Model Explainability**	**Model Serving - FastAPI**	**Github Actions - CI/CD**	**Drift Analysis**	**Results & Interpretation**
Hyperopt, H2O AutoML, and TPOT	SHAP Values Globally & Locally	Create FastAPI, create docker image	Build and Push Docker Images	Generate timestamp data, Drift analysis	Business Conclusion

Data Preparation & Model Build

Model Preparation:

1) Encoding
2) Check Correlations
3) **Advanced Imputation for dealing with missing value - KNN imputer**
4) Remove Outliers
5) Standardization
6) **Split dataset to train, validation, and test**
7) **Leakage Analysis**
8) **Balanced vs Imbalanced Classification Check - SMOTEENN**

Model Building :

1) Check the overfitting for the model best two models, LightGBM & Random Forest.
2) Unsupervised Learning, K-means
3) Tuning Parameters use Hyperopt
4) Try TPOT and H2O Auto ML

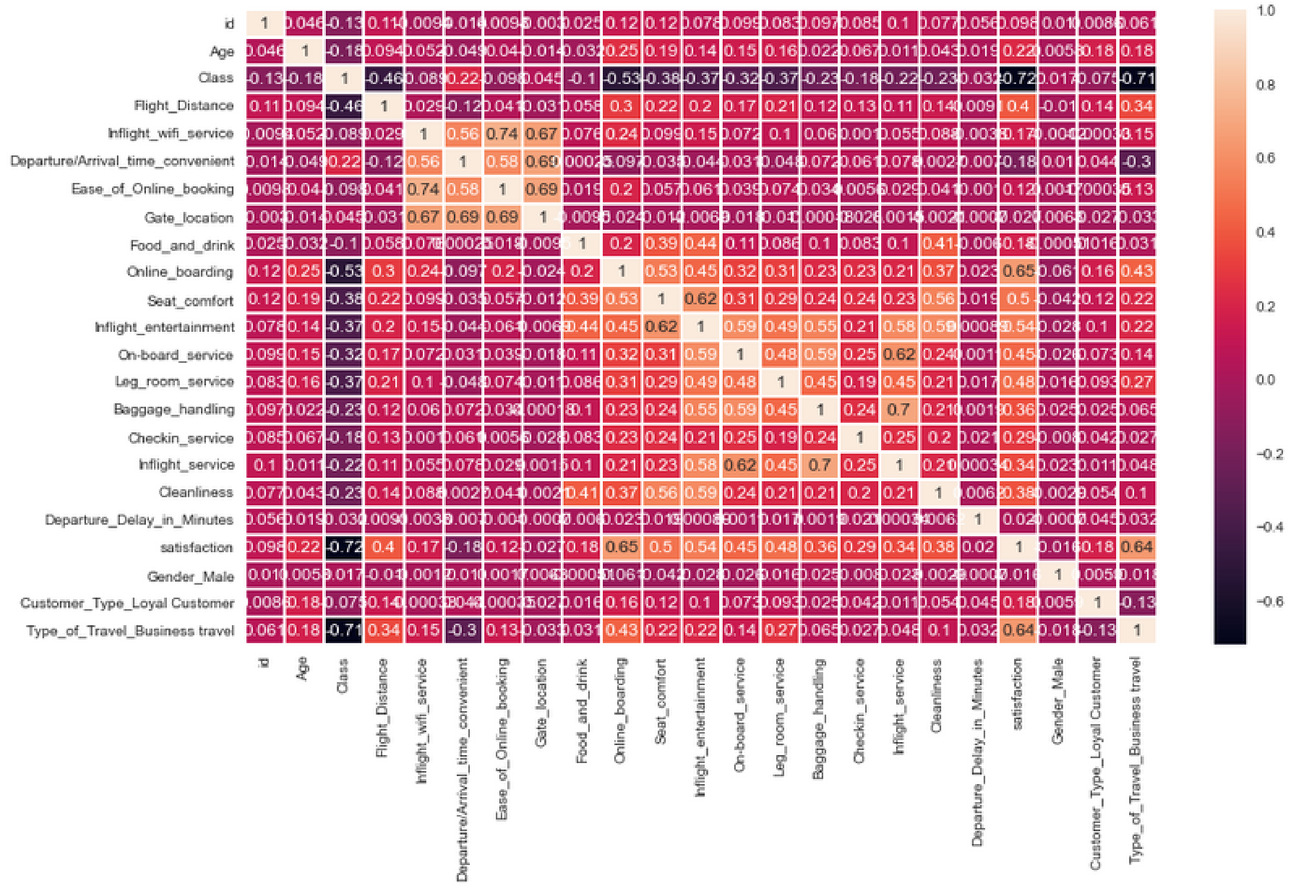
Data Preparation - Leakage Analysis

Data leakage in machine learning refers to the unintended or improper use of data during model training or evaluation that can lead to artificially inflated performance metrics and compromised model integrity, and detecting and mitigating it is crucial for reliable and valid machine learning models.

1) Detection

a) Correlation Coefficient
b) Cross-Validation Accuracy

```
Top 10 - Positive Correlations:
--------------------------------------
Baggage_handling                    0.363072
Cleanliness                         0.378198
Flight_Distance                     0.397443
On-board_service                    0.447795
Leg_room_service                    0.482966
Seat_comfort                        0.499405
Inflight_entertainment              0.535067
Type_of_Travel_Business travel      0.639366
Online_boarding                     0.652242
satisfaction                        1.000000
Name: satisfaction, dtype: float64

Top 10 - Negative Correlations:
--------------------------------
Class                              -0.718454
Departure/Arrival_time_convenient  -0.179432
Gate_location                      -0.026800
Gender_Male                        -0.015567
Departure_Delay_in_Minutes          0.019626
id                                  0.098392
Ease_of_Online_booking              0.120327
Inflight_wifi_service               0.165121
Food_and_drink                      0.175623
Customer_Type_Loyal Customer        0.183222
Name: satisfaction, dtype: float64
```



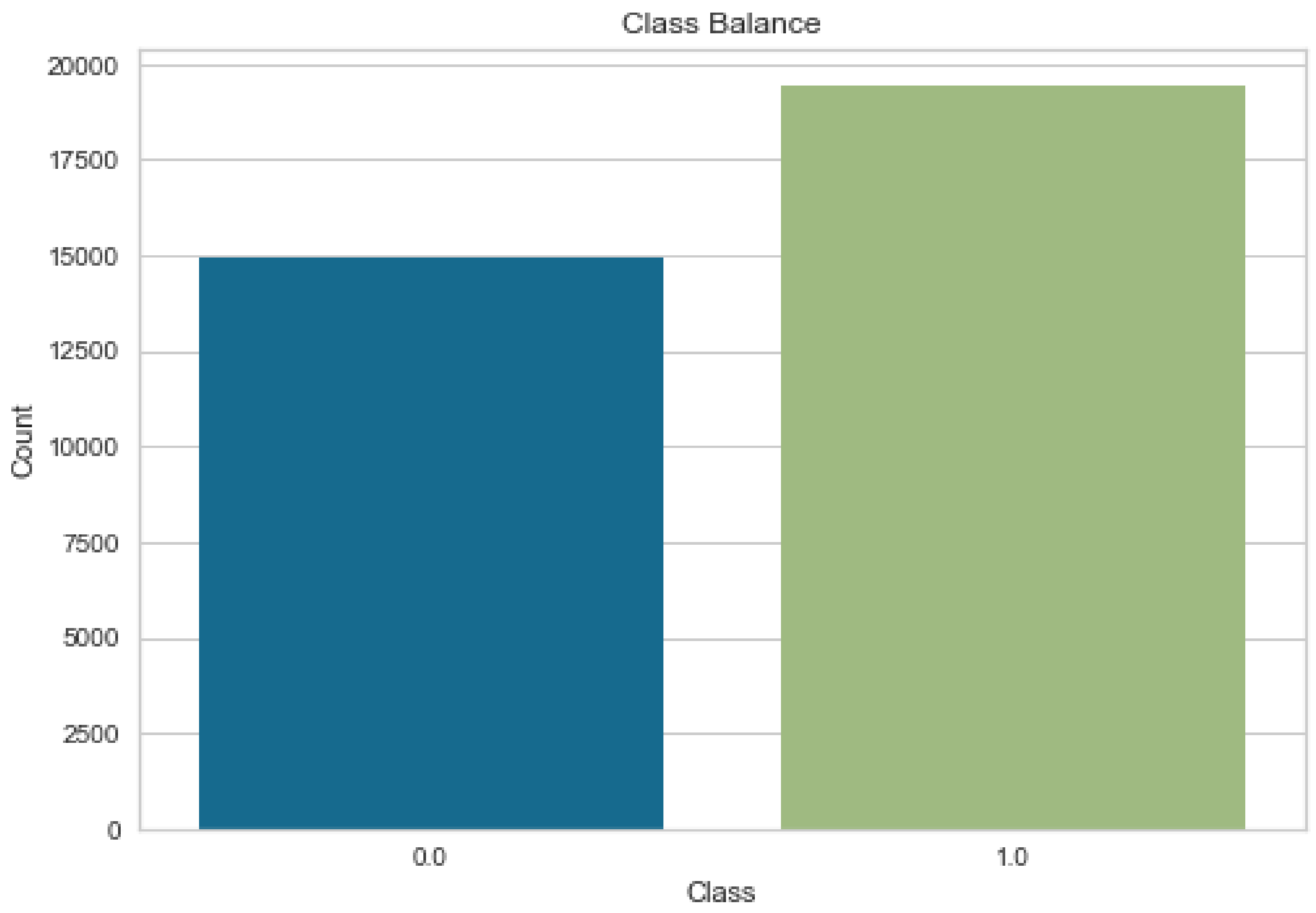

```
Cross-validation accuracy: 0.980100
```

```
Fraction of those non-satisfied women: 0.21
Fraction of those satisfied women: 0.29
Two ratios are similar.
```

2) Solution

a) Remove high correlated features which detected as leaks
b) Create a separate validation set

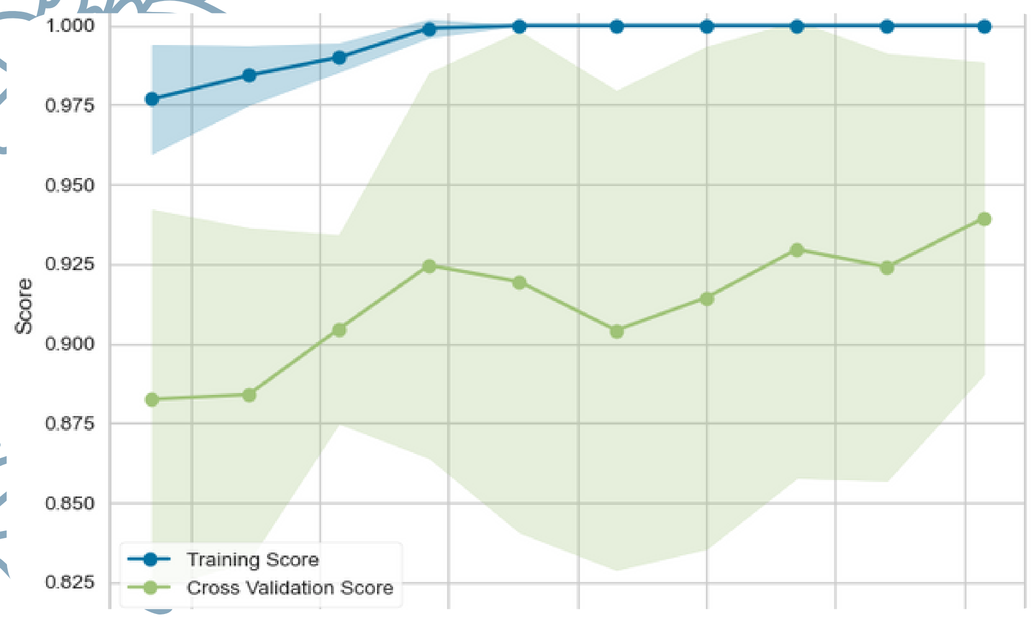
Data Preparation - Imbalanced Data



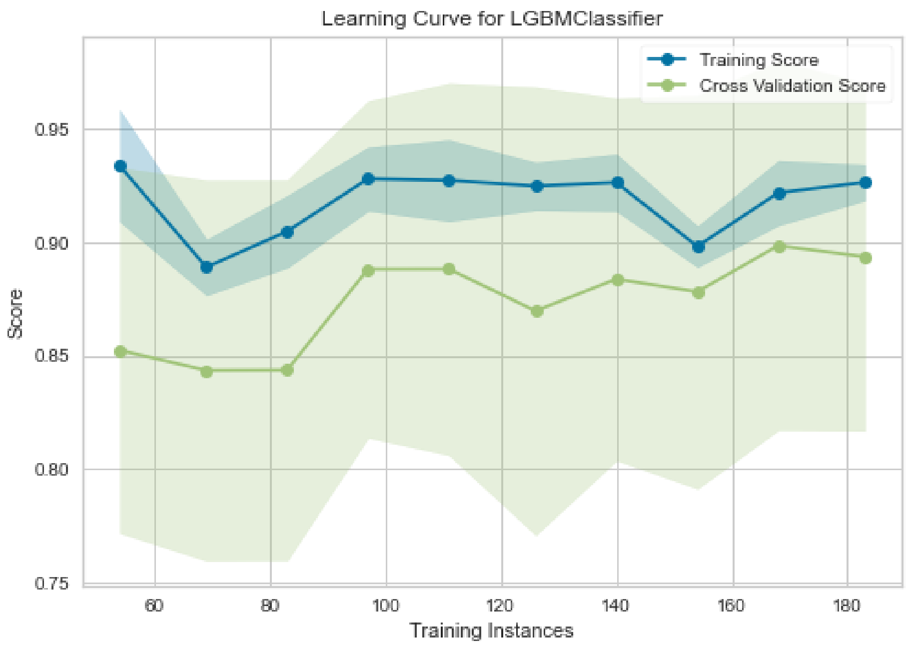
Graphed the frequency of satisfied and non satisfied to see if the dataset is imbalanced:

Our dataset is not much inbalanced, but we still have more cases in which the client is satisfied which means much of satisfied cases.

Model Build - Overfitting Check



LightGBM **by Learning Curve**



• Gridsearchcv

```
{'learning_rate': 0.06, 'max_depth': 10, 'num_leaves': 31} 0.6612352683137512
```

• Bayesian Optimization

iter	target	max_fe...	min_sa...	n_esti...
1	-0.6136	0.2722	16.31	115.1
2	-0.6223	0.806	19.94	75.42
3	-0.6131	0.3485	20.44	240.0
4	-0.6255	0.8875	10.23	130.2
5	-0.6215	0.7144	18.39	98.86
6	-0.6203	0.8021	18.14	230.7
7	-0.6244	0.9241	16.19	80.11
8	-0.6154	0.498	20.94	245.3
9	-0.6208	0.7962	13.79	241.6
10	-0.6155	0.5215	24.8	240.2
11	-0.6183	0.1	22.17	114.9
12	-0.6163	0.3691	11.42	112.6
13	-0.6253	0.999	13.98	119.0
14	-0.618	0.1	16.25	112.1
15	-0.6172	0.1	21.89	242.0

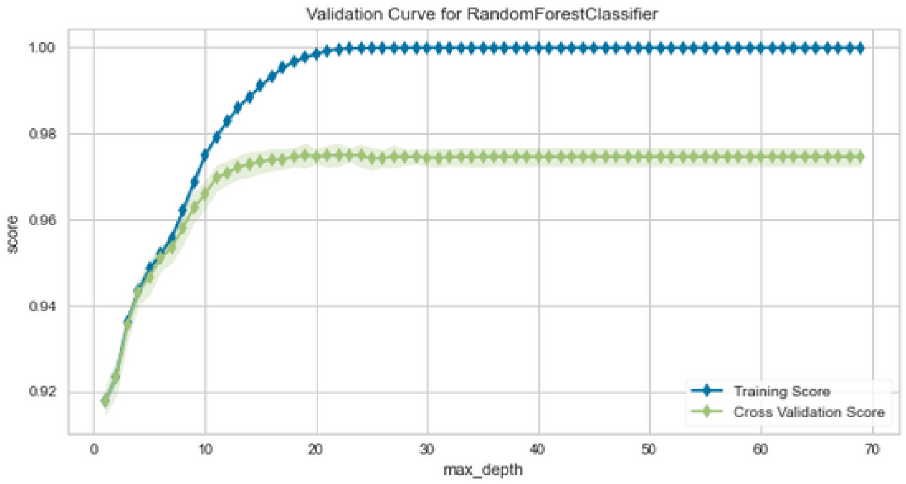
```
Final result: {'target': -0.6130692158901937, 'params': {'max_features': 0.34854136537364394, 'min_samples_split': 2
0.443060083305443, 'n_estimators': 239.95344488408924}}
```

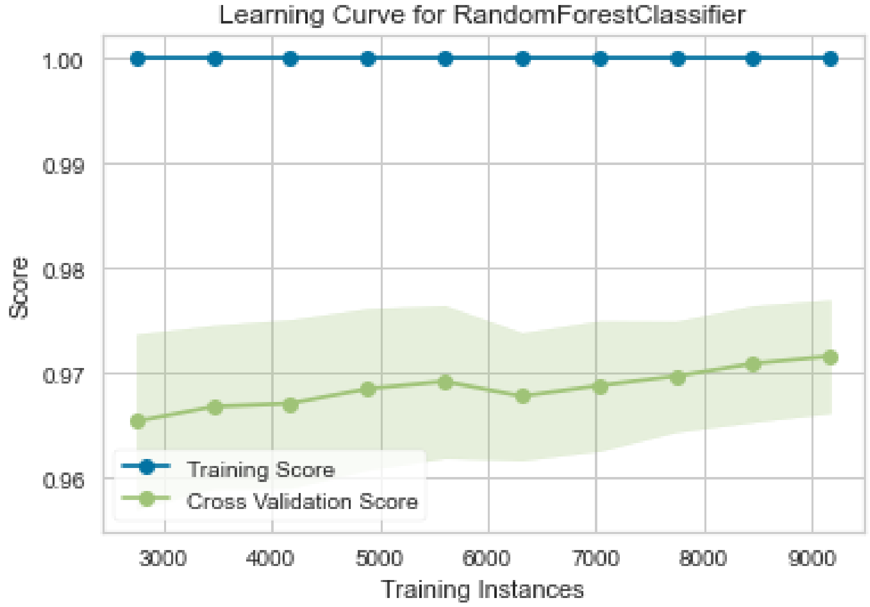
Model Build - Overfitting Check

Random Forest

by Learning Curve & Validation Curve

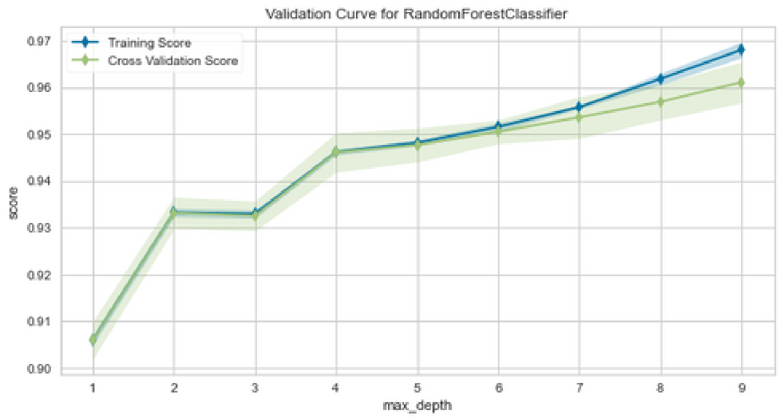
```
{'n_estimators': 1100, 'max_features': 'auto', 'max_depth': 60, 'bootstrap': False}
```

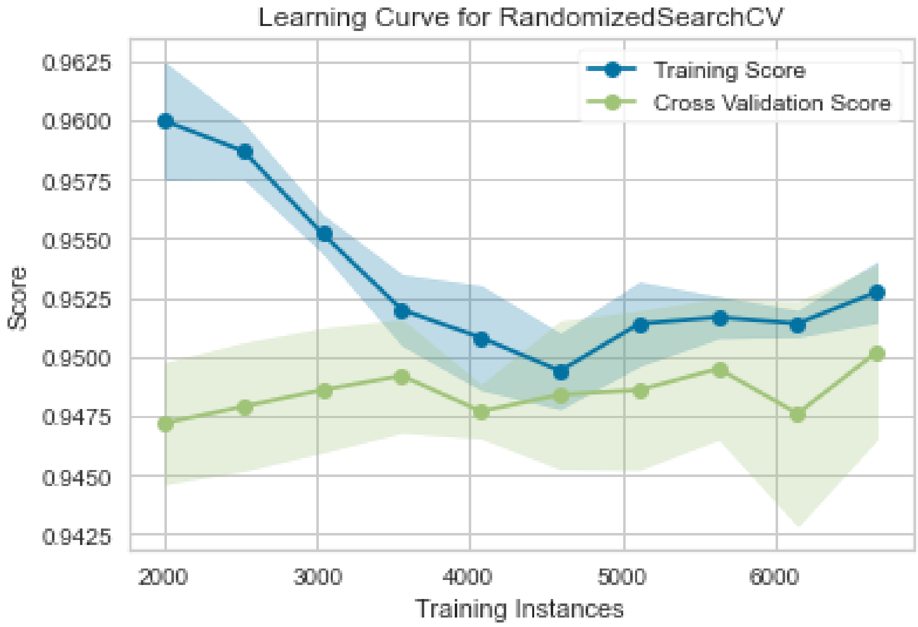




- **Gridsearchcv**

```
{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 5, 'bootstrap': False}
```





Model Build - Model Tuning & AutoML

TPOT

```
Best pipeline: ExtraTreesClassifier(PolynomialFeatures(MinMaxScaler(input_matrix), degree=2, include_bias=False, inte
raction_only=False), bootstrap=False, criterion=entropy, max_features=0.6000000000000001, min_samples_leaf=3, min_sam
ples_split=13, n_estimators=100)
```

```
▾ TPOTClassifier
TPOTClassifier(generations=25, max_time_mins=180, population_size=50,
               scoring='f1', verbosity=2)
```

Model found by TPOT: ExtraTreeClassifier

F1-Score: 0.98

H2O

Best Model: Gradient Boosting Machine

Best Parameters:

Model Summary:

	number_of_trees	number_of_internal_trees	model_size_in_bytes	min_depth	max_depth	mean_depth	min_leaves	max_leaves	mean_leaves
	78	78	216621	10	10	10	100	344	216.564

AUC-ROC: 0.9967

Drawback: Overfitting(Fail to pass the test)

Feature Importance:

Variable Importances:

variable	relative_importance	scaled_importance	percentage
Class	9761.7294921875	1.0	0.3307821004780439
Online_boarding	5815.14697265625	0.5957086781917307	0.1970497678452598
Type_of_Travel_Business travel	3102.840576171875	0.3178576684239343	0.1051416272143226
Inflight_entertainment	2952.356201171875	0.3024419190815216	0.10004237326639631
Inflight_wifi_service	1295.320068359375	0.13269370651953064	0.04389270296274746
Seat_comfort	1245.5628662109375	0.12759653575811392	0.04220665011179076
id	599.2272338867188	0.0613853553682564	0.02030517678730302
Customer_Type_Loyal Customer	591.0216064453125	0.06054476380628233	0.02002712414478912
Leg_room_service	587.2386474609375	0.06015723422073065	0.019898936294482947
Age	552.7769775390625	0.05662695099075021	0.018731181792387603
---	---	---	---
Checkin_service	379.6504211425781	0.03889171703092364	0.012864683850686634
Flight_Distance	377.2779846191406	0.03864868258448295	0.012784292406004465
Baggage_handling	340.9688720703125	0.034929145736234186	0.011553936194535255
Inflight_service	254.11424255371094	0.02603168247563953	0.008610814608269532
On-board_service	241.06507873535156	0.024694914864039266	0.008168635809853376
Departure/Arrival_time_convenient	227.5391845703125	0.023309310583991955	0.007710302715667959
Ease_of_Online_booking	86.67536926269531	0.008879099685364492	0.0029370472442788054
Food_and_drink	46.52534866333008	0.004766096899178082	0.0015765395433920179
Departure_Delay_in_Minutes	43.2326774597168	0.004428792817329833	0.0014649653906984 36
Gender_Male	34.103919982910156	0.0034936350172583845	0.0011556319293123756

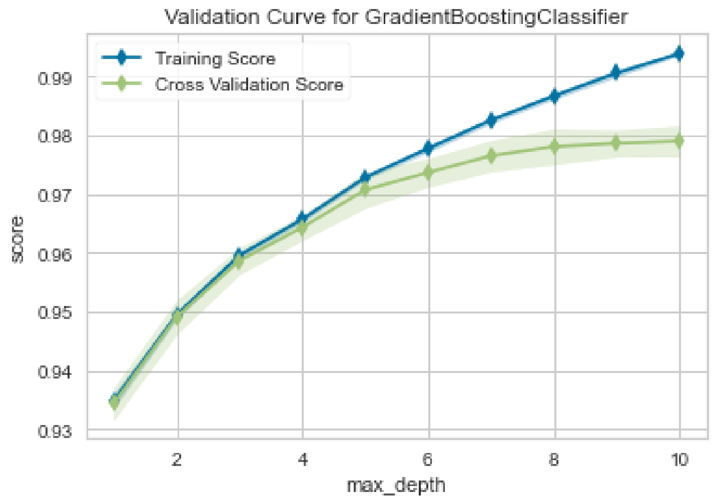
[22 rows x 4 columns]

Model Build - Best Model by Hyperopt with LightGBM

```
space_eval(search_space, best_params)
```

```
{'learning_rate': 0.017868454799152795,
 'max_depth': 6,
 'n_estimators': 689,
 'n_jobs': -1,
 'num_leaves': 54,
 'random_state': 42,
 'verbose': -1}
```

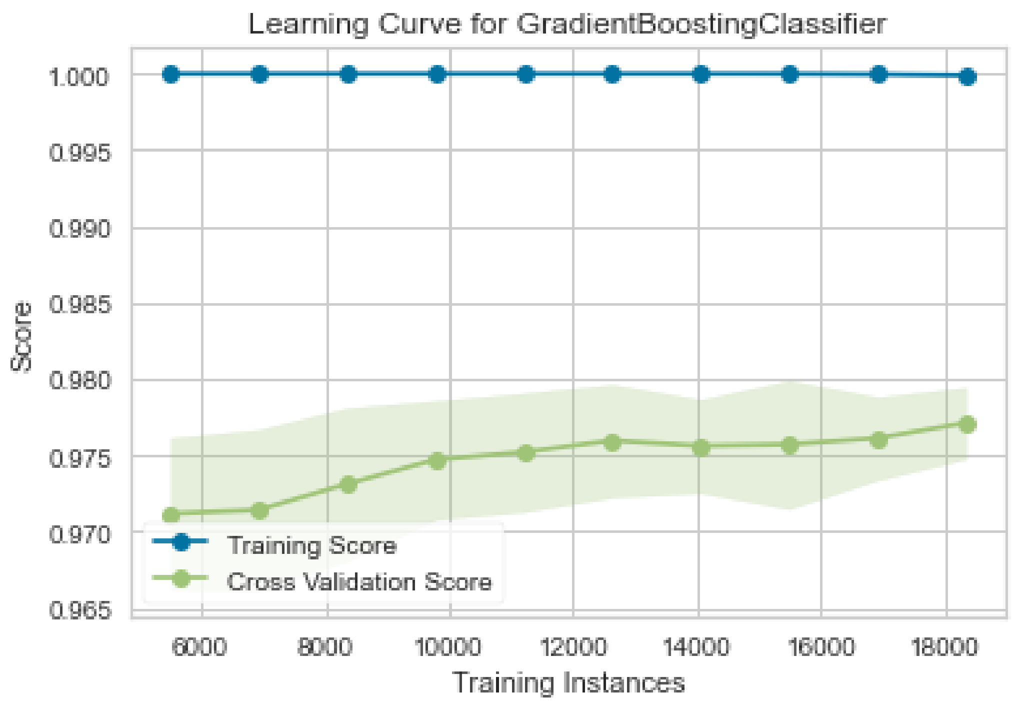
- Best Parameter Output provided by HyperOPT
- Compare with GridsearchCV, HyperOPT can reduce time, labour and computing costs



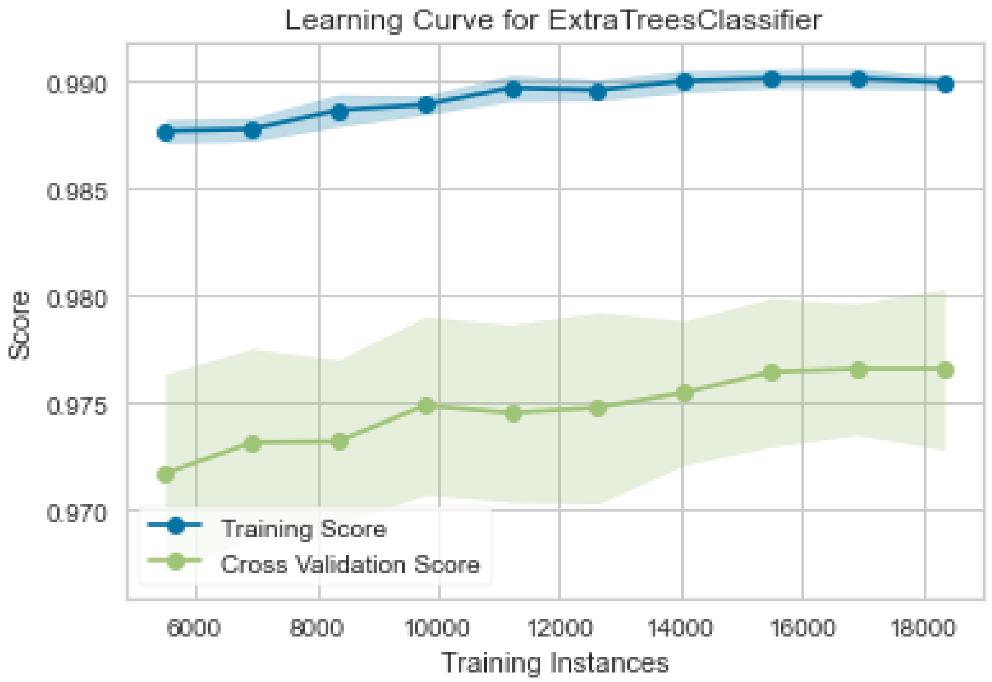
Accuracy Comparsion

	precision	recall	f1-score	support
GBM	0.983823	0.973053	0.978408	2375
ExtraTree	0.985507	0.973474	0.979454	2375
LightGBM	0.983823	0.973053	0.978408	2375

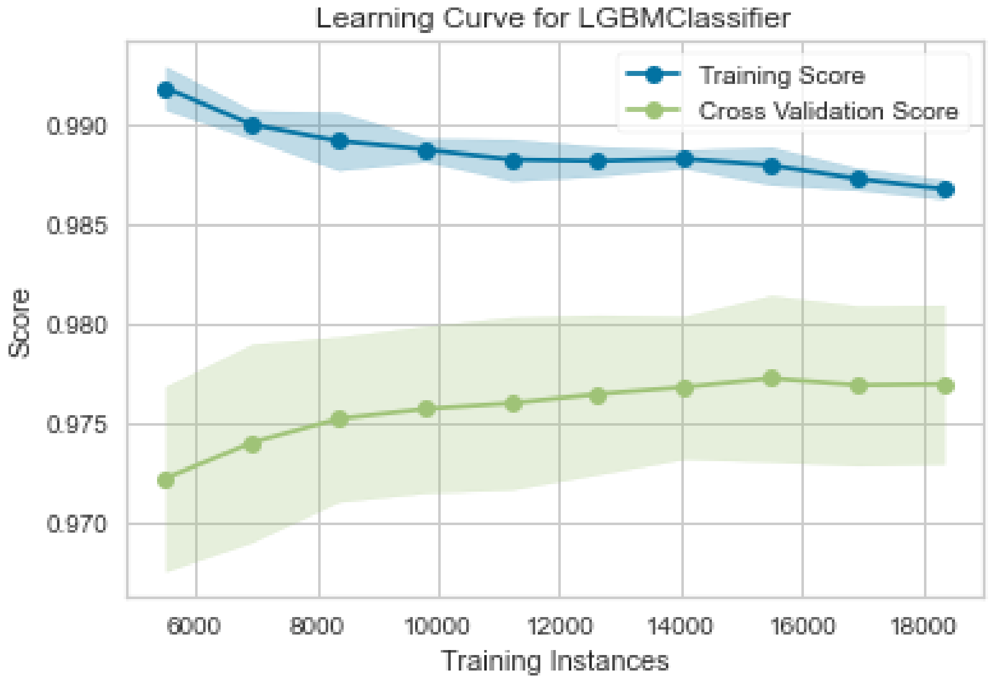
H2O-GradientBoosting



TPOT-ExtraTrees

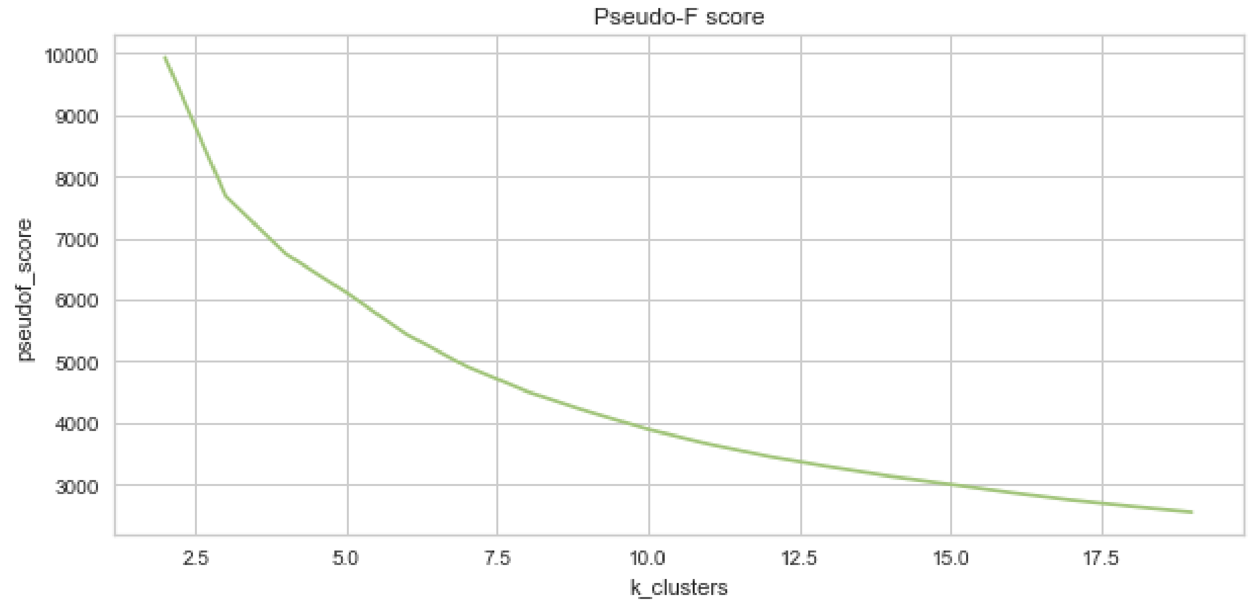


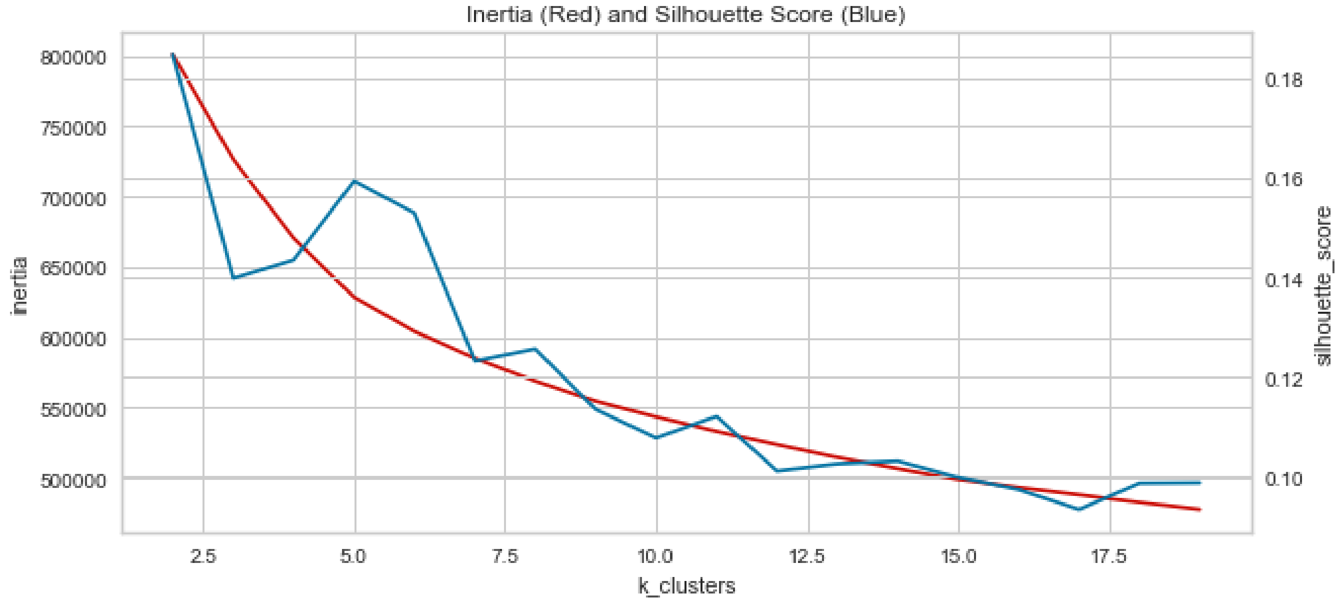
Hyperopt-LightGBM



Model Build - Unsupervised Learning

by K-means Clustering





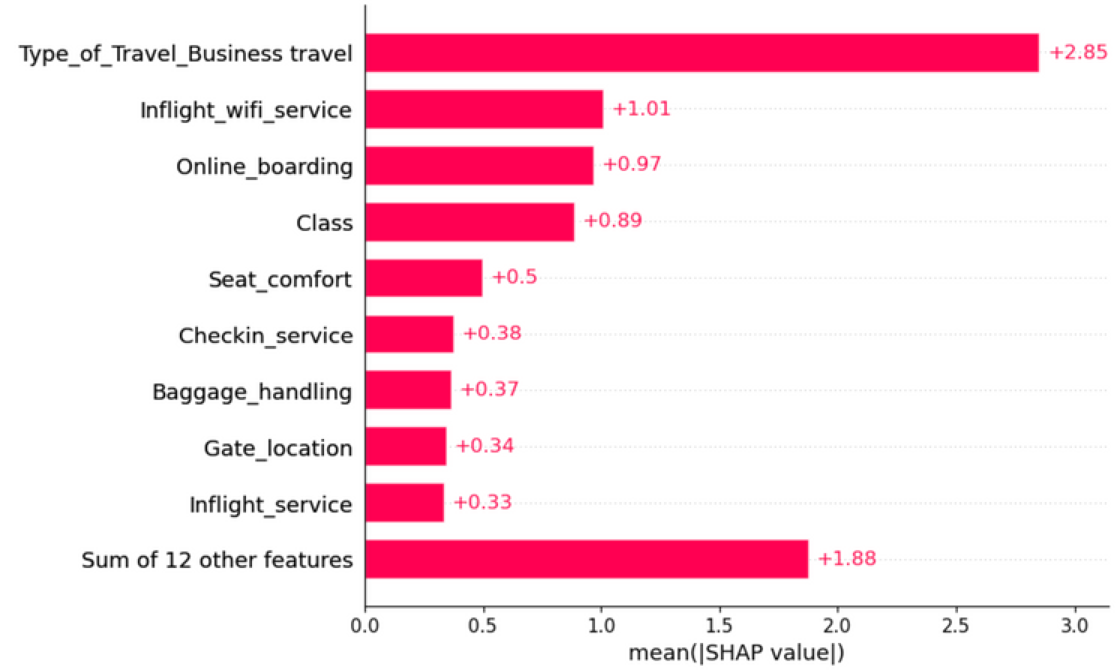
The Elbow method suggests we should pick k = 3 to fit the optimal clustering model.

	id	Age	Class	Flight_Distance	Inflight_wifi_service	Departure/Arrival_time_convenient	Ease_of_Online_booking	Gate_location
0	74705.424663	0.223326	0.044534	0.429665	1.704909	1.644152	1.794614	1.598452
1	72781.171262	0.189999	0.144920	0.324029	3.879028	3.832911	3.788779	3.829285
2	64310.050700	-0.258964	1.465793	-0.466051	2.645630	3.370916	2.686519	2.981332

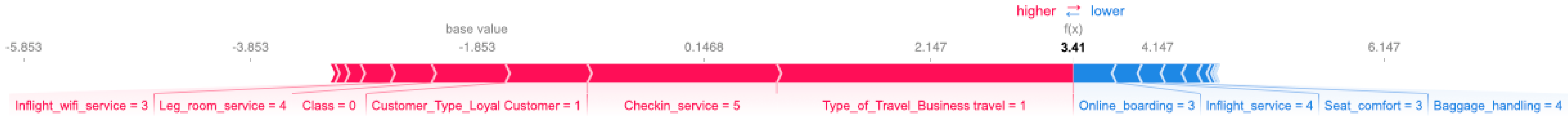
Model Explainability & Interpretability

by SHAP

Global Analysis



Local Analysis



Mean values of features
in training dataset

Age	-0.002344
Class	0.698960
Flight_Distance	0.003636
Inflight_wifi_service	2.856910
Departure/Arrival_time_convenient	3.146674
Ease_of_Online_booking	2.865447
Gate_location	2.967891
Food_and_drink	3.434049
Online_boarding	3.641706
Seat_comfort	3.844614
Inflight_entertainment	3.790158
On-board_service	3.762420
Leg_room_service	3.679642
Baggage_handling	3.957606
Checkin_service	3.601731
Inflight_service	3.966930
Cleanliness	3.603217
Departure_Delay_in_Minutes	0.000859
Gender_Male	0.499927
Customer_Type_Loyal Customer	0.950846
Type_of_Travel_Business travel	0.754553

features	**compare with the mean value**	**magtinude**
check in service	better	large
leg room service	slightly better	medium
inflight wifi service	slightly better	small
On board service	worse	medium

Model Serving - Overview

by FastAPI and Docker Container

LightGBM

```python
lgbm_clf = LGBMClassifier(n_estimators=689,
                          n_jobs=-1,
                          learning_rate=0.017868454799152795,
                          num_leaves=54,
                          max_depth=6,
                          verbose=-1,
                          random_state=42)
model_lgbm= lgbm_clf.fit(X_train, y_train)
```

Step 1: Select the final model - LightGBM with tuned parameters is chosen as the final model

```python
import pickle
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
from lightgbm import LGBMClassifier
```

```python
# Define the preprocessor for the specified columns
preprocessor = ColumnTransformer([
    ('scaler', StandardScaler(), ['Flight_Distance', 'Departure_Delay_in_Minutes'])
])

# Define the pipeline with specified hyperparameters for the LightGBM model
pipeline = Pipeline([
    ('preprocessor', preprocessor),  # Standardize selected columns
    ('lgbm', LGBMClassifier(n_estimators=689,
                            n_jobs=-1,
                            learning_rate=0.017868454799152795,
                            num_leaves=54,
                            max_depth=6,
                            verbose=-1,
                            random_state=42))  # Train a LightGBM model with specified hyperparameters
])
```

Step 2: Pickle the model with all data preprocessing steps

FASTAPI
- app
  - __pycache__
  - __init__.py
  - main.py
  - trained_pipeline_v1.0.0.pkl
- Dockerfile
- requirements.txt

app > main.py > ...

```python
from fastapi import FastAPI
from pydantic import BaseModel
import pickle
import pandas as pd
from pathlib import Path


# Define the FastAPI app
app = FastAPI()

# Define the input schema - need to rename column names in the model code
class InputData(BaseModel):
    Class: int
    Flight_Distance: float
    Inflight_wifi_service:int
    Departure_Arrival_time_convenient: int
    Ease_of_Online_booking: int
    Online_boarding: int
    Seat_comfort: int
    Inflight_entertainment: int
    On_board_service: int
    Leg_room_service: int
    Baggage_handling: int
    Checkin_service: int
    Inflight_service: int
    Cleanliness: int
    Departure_Delay_in_Minutes: float
    Customer_Type_Loyal_Customer: int
```

Step 3: Create the FastAPI app with the pickled model

Dockerfile ×

Dockerfile > ...

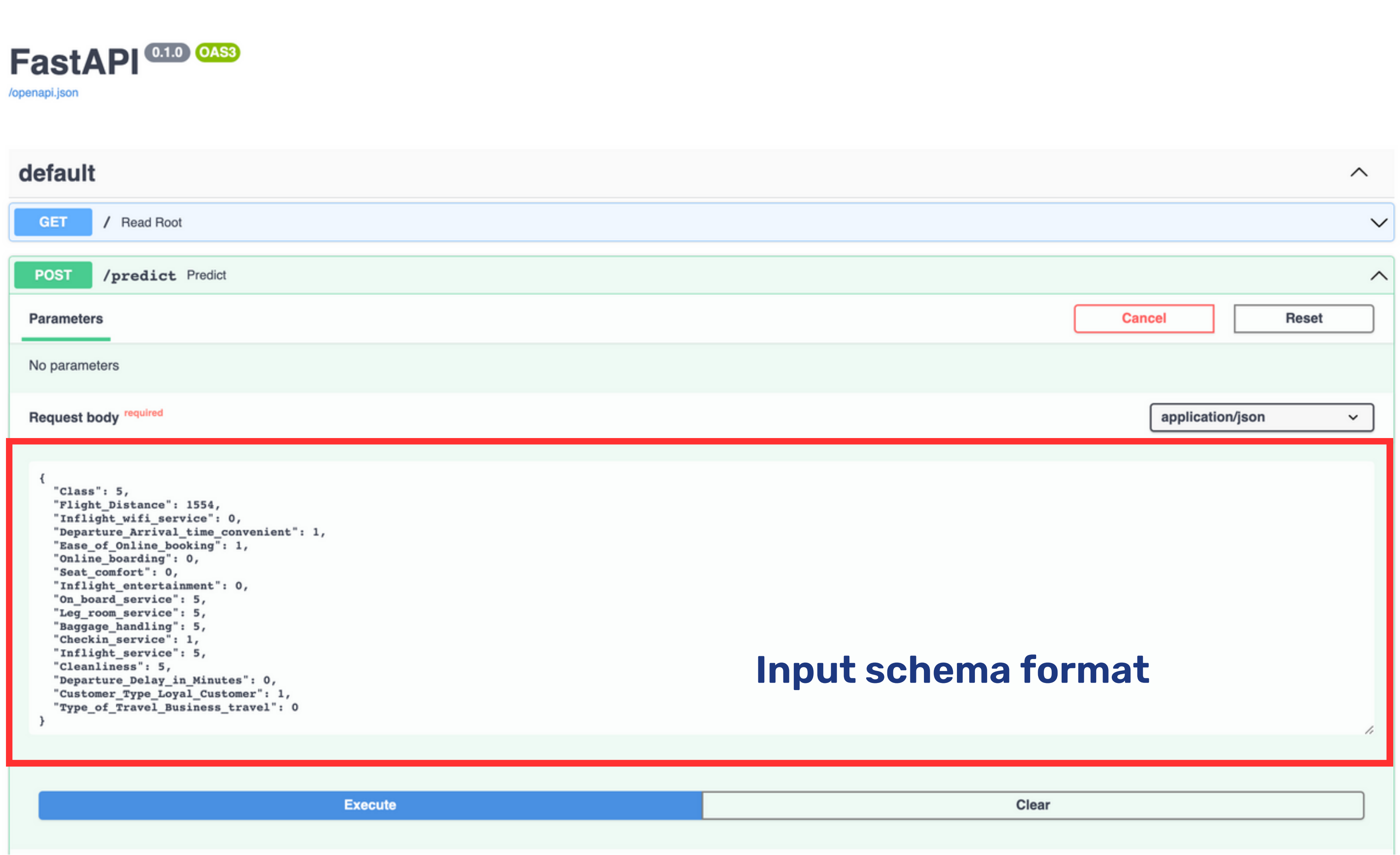
```dockerfile
FROM tiangolo/uvicorn-gunicorn:python3.10

LABEL maintainer="Sebastian Ramirez <tiangolo@gmail.com>"

COPY requirements.txt /app/requirements.txt
RUN pip install --no-cache-dir -r /app/requirements.txt

COPY ./app /app/app
```

Step 4: Create Use docker build and run commands to build and run your image, either locally or on a cloud service

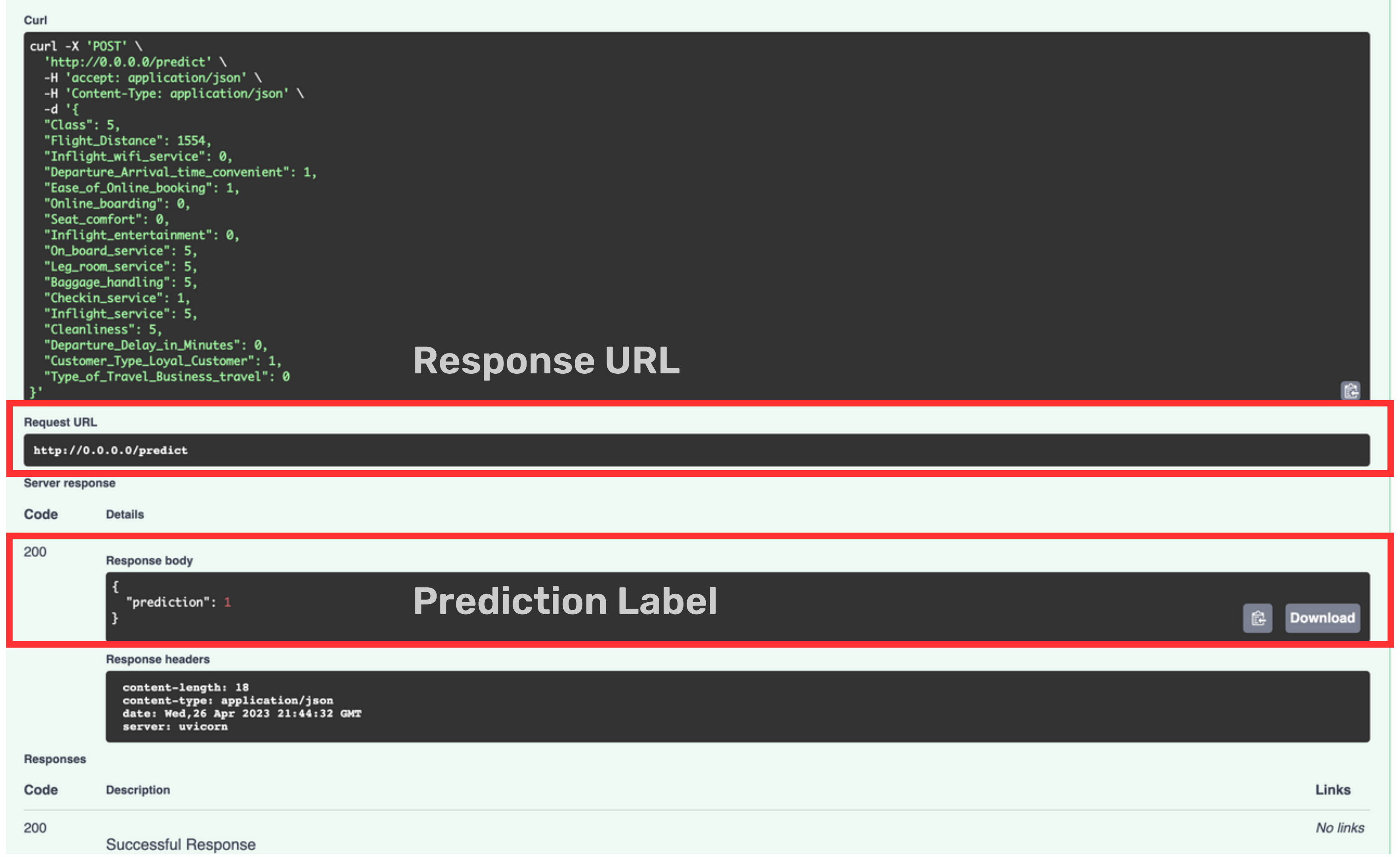
Model Serving - Results

By FastAPI and Docker Container

Post Data



Prediction Response



Github Actions - CI/CD Docker Hub

Github Workflows - YAML

```
name: Python application with Docke

on:
  push:
    branches:
      - main
```

```
- name: Set up Docker Buildx
  uses: docker/setup-buildx-action@v1

- name: Login to Docker Hub
  uses: docker/login-action@v1
  with:
    username: ${{ secrets.DOCKER_LOGIN_USERNAME }}
    password: ${{ secrets.DOCKER_LOGIN_PASSWORD }}
```

```
- name: Build and push Docker image
  uses: docker/build-push-action@v2
  with:
    context: ./FASTAPI
    push: true
    tags: nishivijan06/enterprise1:tagname #${{ secrets.TEST_SECRET }}:tagname

- name: Lint with flake8
  run: |
    flake8 . --count --select=E9,F63,F7,F82 --show-source --statistics
    flake8 . --count --exit-zero --max-complexity=10 --max-line-length=127 --st
```

Step 1: On every **push** on the main branch, action is **triggered**

Step 2: Setup buildx and login into Docker Hub

Step 3: Build and Push Docker Image

Workflow Automation

- Automate the CI/CD workflow
- Docker Hub - Host and manage docker images
- Separate the container management from the deployment service
- Automatically pulls our code from GitHub or BitBucket, locates the Dockerfile in it, and starts building, tagging, and pushing the image into the container.

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
tagname		Image	---	10 hours ago

Docker Image is created in the Dockerhub on trigger

Github Actions - CI/CD HEROKU

Github Workflows - YAML

```yaml
e: Deploy to heroku.

un workflow on every push to main bra

ush:
 branches: [main]
```

Step 1: On every **push** on the main branch, action is **triggered**

```yaml
name: Build, Push and Release a Docker container to Heroku.
uses: gonuit/heroku-docker-deploy@v1.3.3 # GitHub action nai
with:
  # Below you must provide variables for your Heroku app.

  # The email address associated with your Heroku account.
  # If you don't want to use repository secrets (which is r
  # email: my.email@example.com
  email: ${{ secrets.HEROKU_EMAIL }}

  # Heroku API key associated with provided user's email.
  # Api Key is available under your Heroku account settings
  heroku_api_key: ${{ secrets.HEROKU_API_KEY }}
```

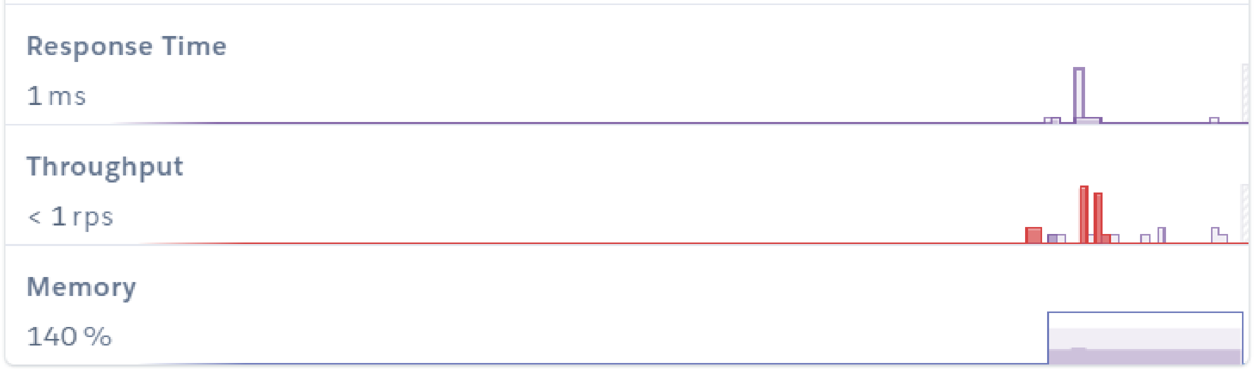
Step 2: Build connection with Heroku

```yaml
dockerfile_directory: ./FASTAPI

# (Optional, default: "Dockerfile")
# Dockerfile name.
dockerfile_name: Dockerfile

# (Optional, default: "")
# Additional options of docker build command.
docker_options: "--no-cache"

# (Optional, default: "web")
# Select the process type for which you want the
# By default, this argument is set to "web".
# For more information look at https://devcenter
process_type: web
```

Step 3: Build and Push to Heroku

Workflow Automation - Heroku

- Container-based cloud Platform as a Service (PaaS)
- Deploy, manage, and scale apps



Docker Image is created in Heroku

Data Drift Reports

Generating Synthetic Data to observe data drift

- CTGAN package was used to generate synthetic data.
- It was preferred because it can observe the patterns in the given dataset to generate both continuous and categorical variables.
- The trained model as pickle is used to generate data daily.

Why detect data drift?

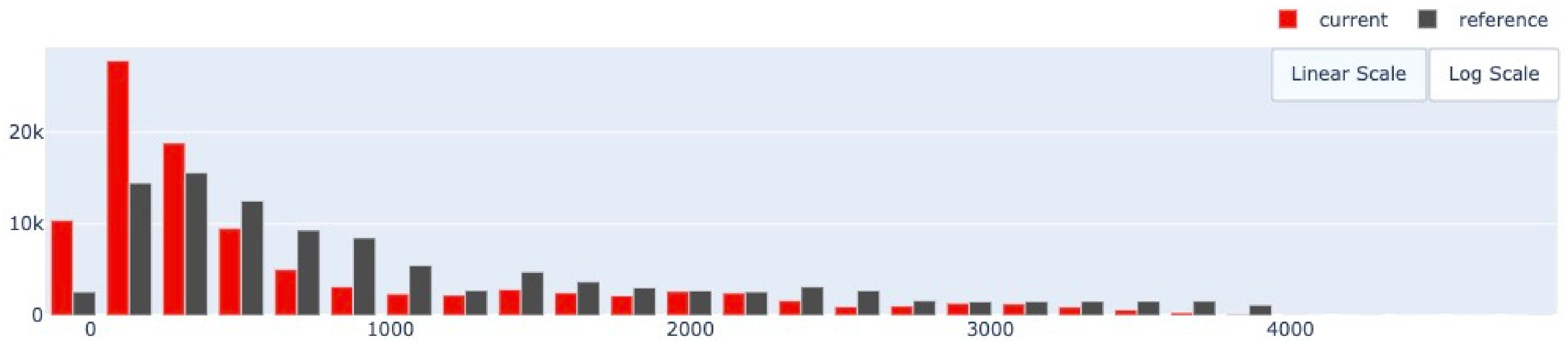
Data drift detection is important for both synthetic and original datasets because it helps us ensure that the data being used for modeling and analysis is still representative of the real-world phenomenon that it is meant to represent.

How to detect data drift

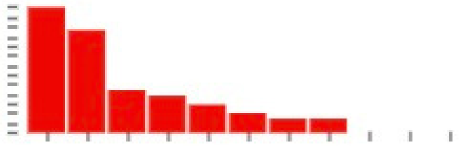
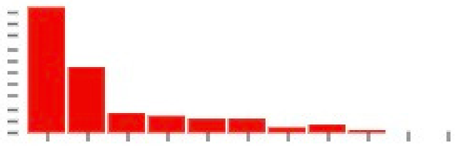
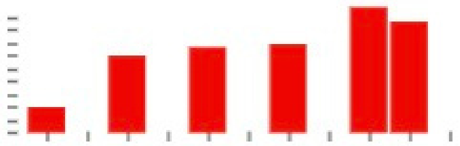
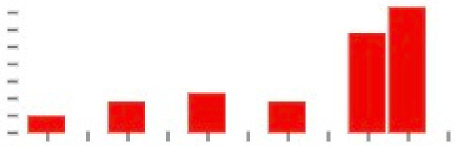
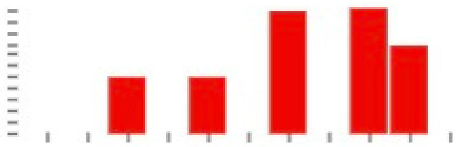
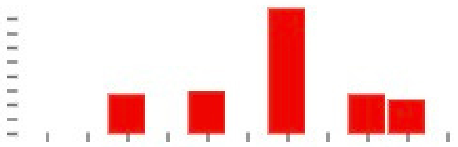
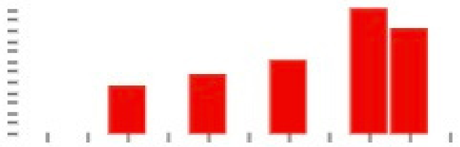
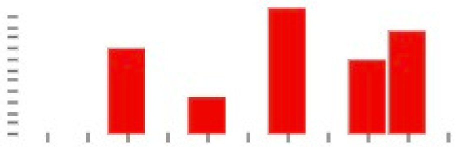
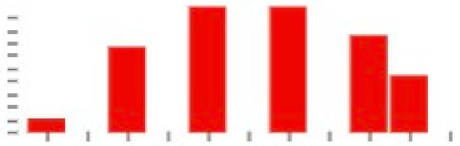
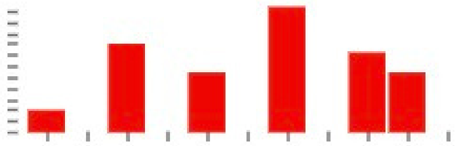
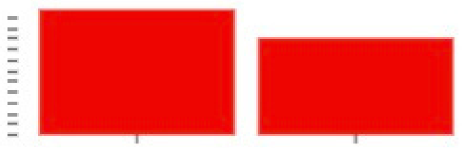
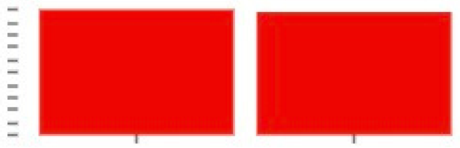
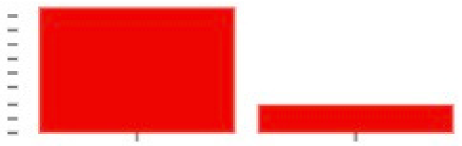
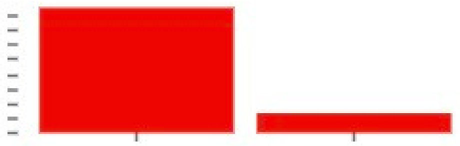
To properly detect data drift in a variable, we need to use statistical tests and machine learning algorithms that can detect differences between the distributions of the variable in the two datasets.

Flight Distance - Distribution & Stat Property

	current	reference
count	100000	103904
mean	809.22	1189.45
std	874.6	997.15
min	-58.0	31.0
25%	220.0	414.0
50%	421.0	843.0
75%	1090.0	1743.0
max	4316.0	4983.0
unique	3980 (3.98%)	3802 (3.66%)
most common	167.0 (0.2%)	337.0 (0.64%)
missing	0 (0.0%)	0 (0.0%)
infinite	0 (0.0%)	0 (0.0%)

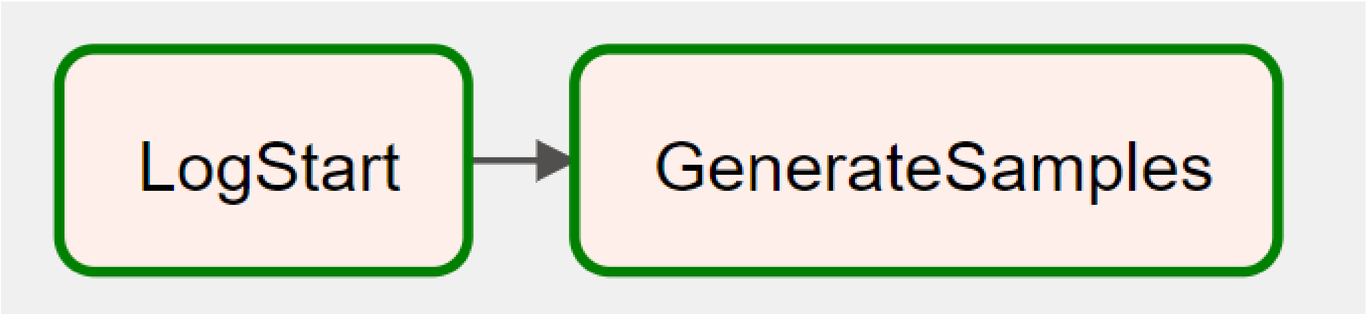


Data Drift Report

	Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
>	Flight Distance	num			Detected	Wasserstein distance (normed)	0.381321
>	Departure/Arrival time convenient	num			Detected	Wasserstein distance (normed)	0.334093
>	Checkin service	num			Detected	Wasserstein distance (normed)	0.285991
>	Seat comfort	num			Detected	Wasserstein distance (normed)	0.207027
>	Inflight wifi service	num			Detected	Wasserstein distance (normed)	0.137727
>	satisfaction	cat			Not Detected	Jensen-Shannon distance	0.044417
>	Customer Type	cat			Not Detected	Jensen-Shannon distance	0.04103

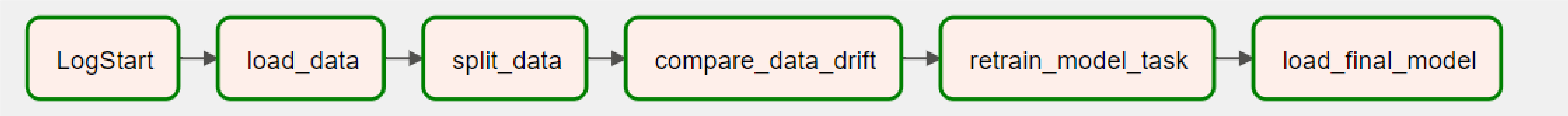
Dynamic ML Monitoring

Generating Data

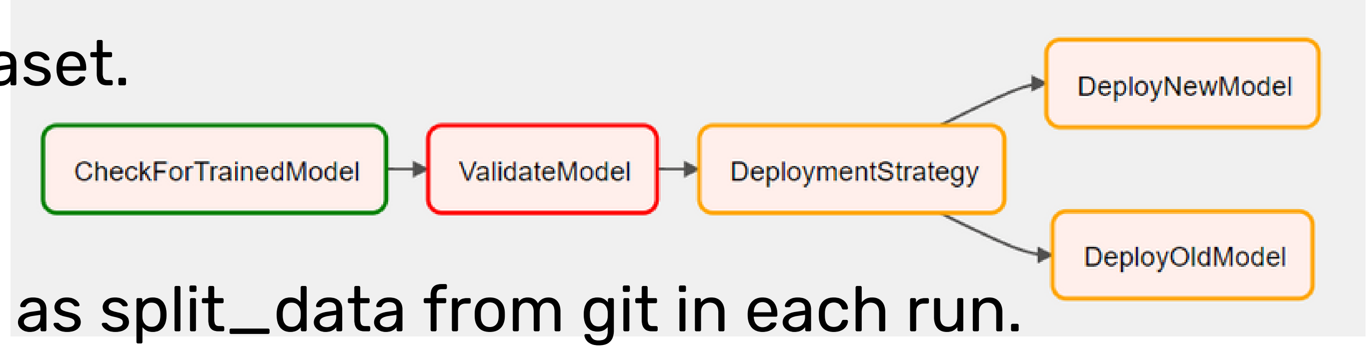


- CTGAN package was was loaded to generate data.
- Data was gen. manually for last 300 days with timestamp for generating historical data pushed into the database.
- Although we are adding timestamp , seasonality can't be recorded.
- **Frequency**: Daily **Destination**: SQL Server

Why detect data drift?



- Data split considered to detect drift in data. Data of last 60 days was compared with data BW last 60 to 120 days.
- Attributes considered for detecting data drift:
  High Feature Importance: inflight_wifi_service, class, seat_comfort
  High sensitivity over time: Flight distance, Arrival time convenient
- If the drift in any column was detected, the model was retrained on new dataset.
- The new model was pickled and stored with timestamp



CI/CD Scope in the pipeline : using python operator to fetch code for tasks such as split_data from git in each run.
Scope of Comparison of Models and Dynamic feature importance : final generated model can be compared to previously generated models in recent data, due to resource constraints, OS was killing the task .

Landing

- Quick Win - Provide better inflight WIFI and service, more convenient online boarding process, more customer-friendly check-in services, more comfortable seats, and more careful luggage handling
- Customers taking business travels in business cabins are prone to give more positive feedbacks; therefore, targeting businessmen with customized ads, discounts, and coupons may be a good idea.
- Semi-supervised learning can save cost of expensive NPS programs
- Focus on operational efficiency increasing satisfaction on short distance flights

Thank you!
Any Question?

Modelling Approach

Pre-processing

- Missing Value Imputation - Custom Function
- Scaling - Standard Scaler
- Test Train Validation Split - 10/80/10%
- Semi-supervised learning

Feature Importances

- Recursive Feature Elimination (RFE)
- Lasso Regression

Model Building

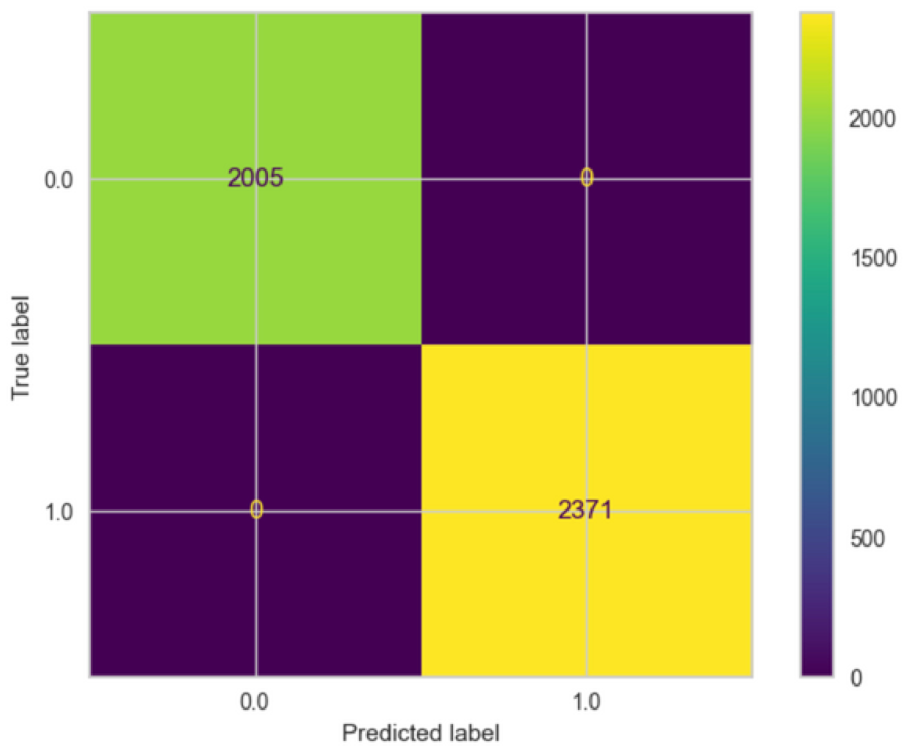
- Logistic Regression
- Random Forest
- Light GBM
- XGBoost
- AdaBoost
- Naive Bayes
- Label Propagation RBF
- Label Spreading RBF

Results Validation

- ROC curve, PR curve
- Classification score
- Confusion Matrix
- Precision, Recall, F1 Score

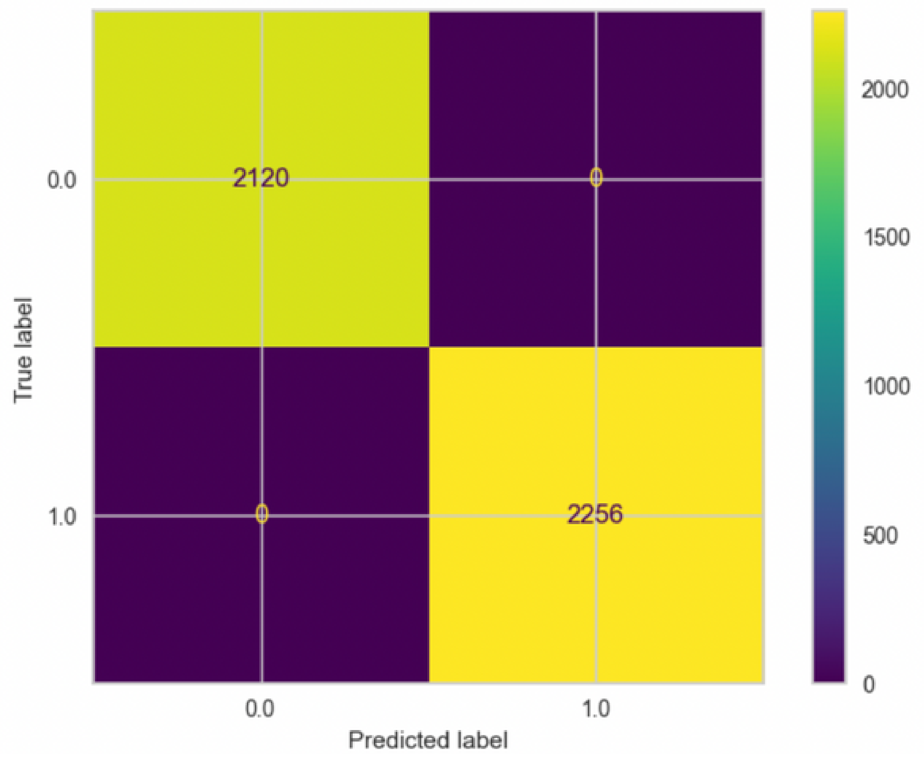
Results

Logistic Regression



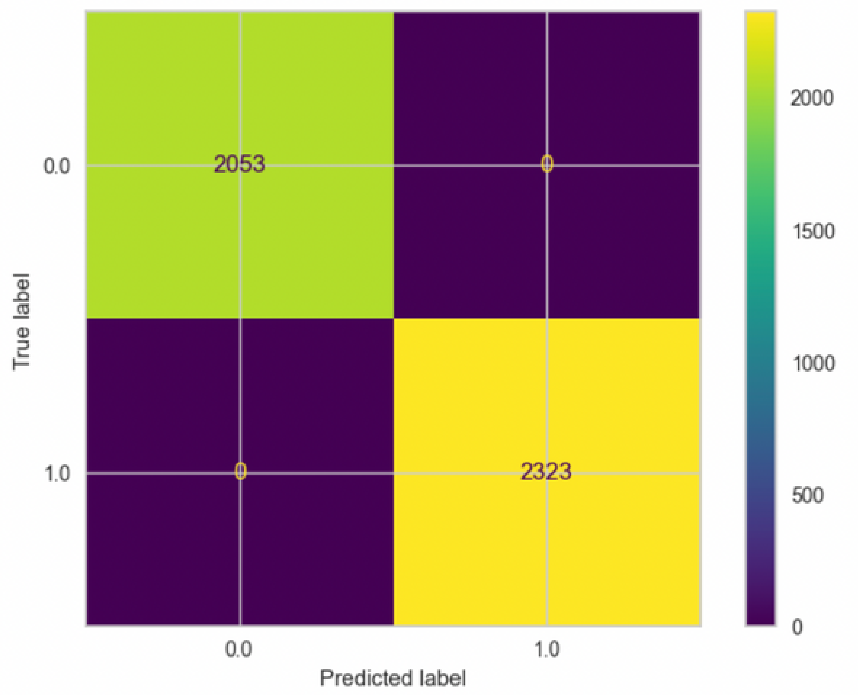
Precision : 93.71%
Recall: 94.41%

Naive Bayes



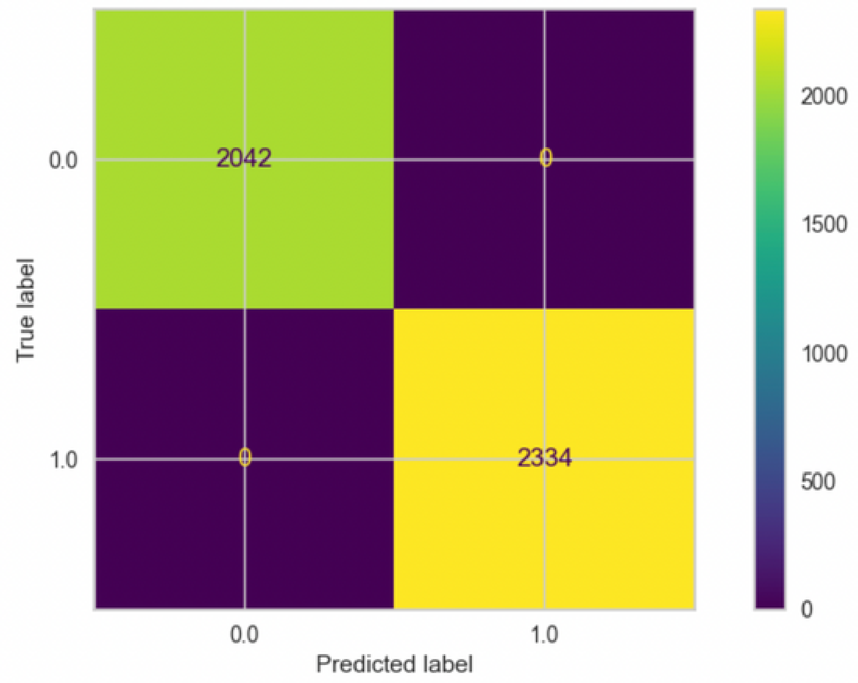
Precision : 94.01%
Recall: 90.44%

Random Forest



Precision : 98.14%
Recall: 97.22%

Light GBM



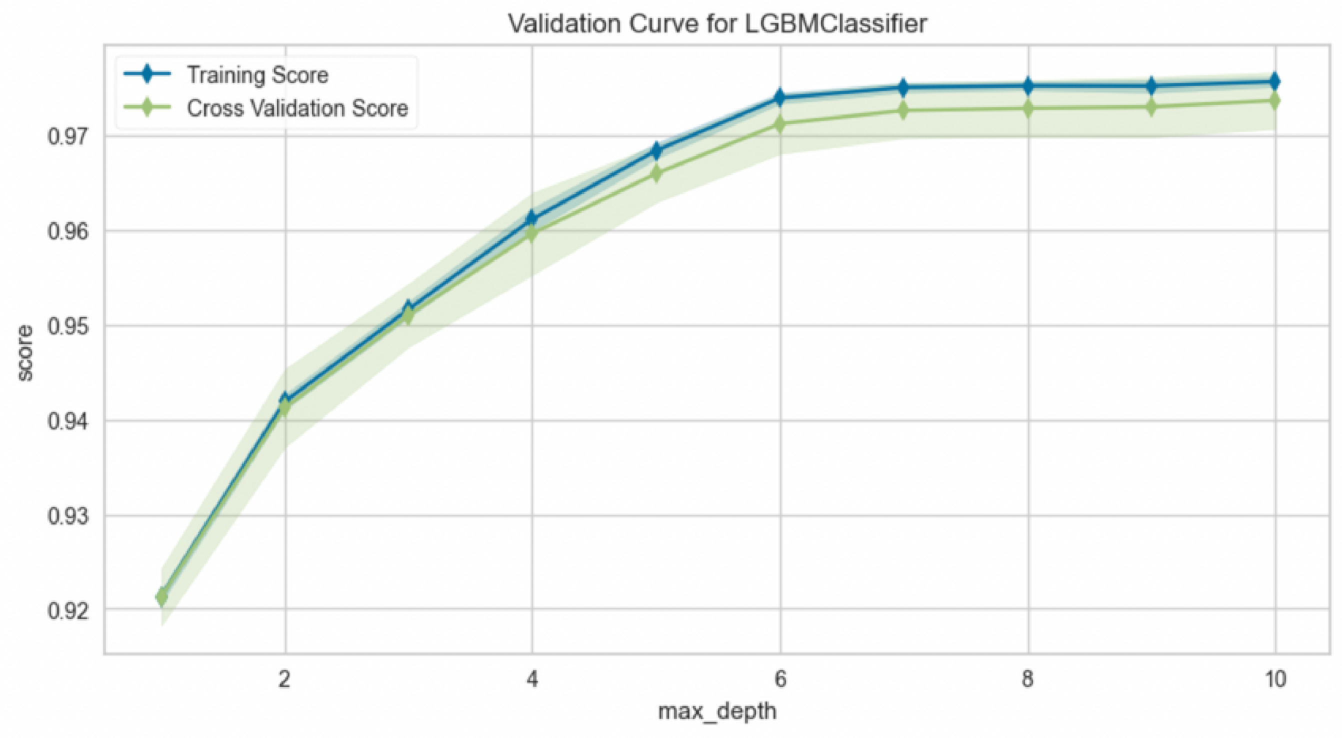
Precision : 98.11%
Recall: 97.65%

Best Model - Supervised Learning

Results - LightGBM				
	precision	recall	f1-score	support
DummyClassifier	0.535878	1.000000	0.697813	2345
LogisticRegression	0.933783	0.944136	0.938931	2345
KNN	0.968830	0.967591	0.968210	2345
Naive Bayesian	0.940160	0.904478	0.921973	2345
Decision Tree	0.973493	0.971002	0.972246	2345
Random Forest	0.981489	0.972281	0.976864	2345
LightGBM	0.981148	0.976546	0.978842	2345
SVM	0.961961	0.970576	0.966249	2345
AdaBoost	0.934379	0.965458	0.949664	2345

Based on the F1 score, LightGBM is the best model

LightGBM also performs really well on the test set



Supervised vs Semi-Supervised Learning

Semi-supervised Learning

	precision	recall	f1-score	support
DummyClassifier	0.535878	1.000000	0.697813	2345
LogisticRegression	0.933783	0.944136	0.938931	2345
KNN	0.968830	0.967591	0.968210	2345
Naive Bayesian	0.940160	0.904478	0.921973	2345
Decision Tree	0.973493	0.971002	0.972246	2345
Random Forest	0.981489	0.972281	0.976864	2345
LightGBM	0.981148	0.976546	0.978842	2345
SVM	0.961961	0.970576	0.966249	2345
AdaBoost	0.934379	0.965458	0.949664	2345
LabelPropagation	0.923297	0.965032	0.943703	2345
Label Spreading	0.941752	0.944563	0.943155	2345

Label Propogation is the best semi-supervised technique

Supervised Learning

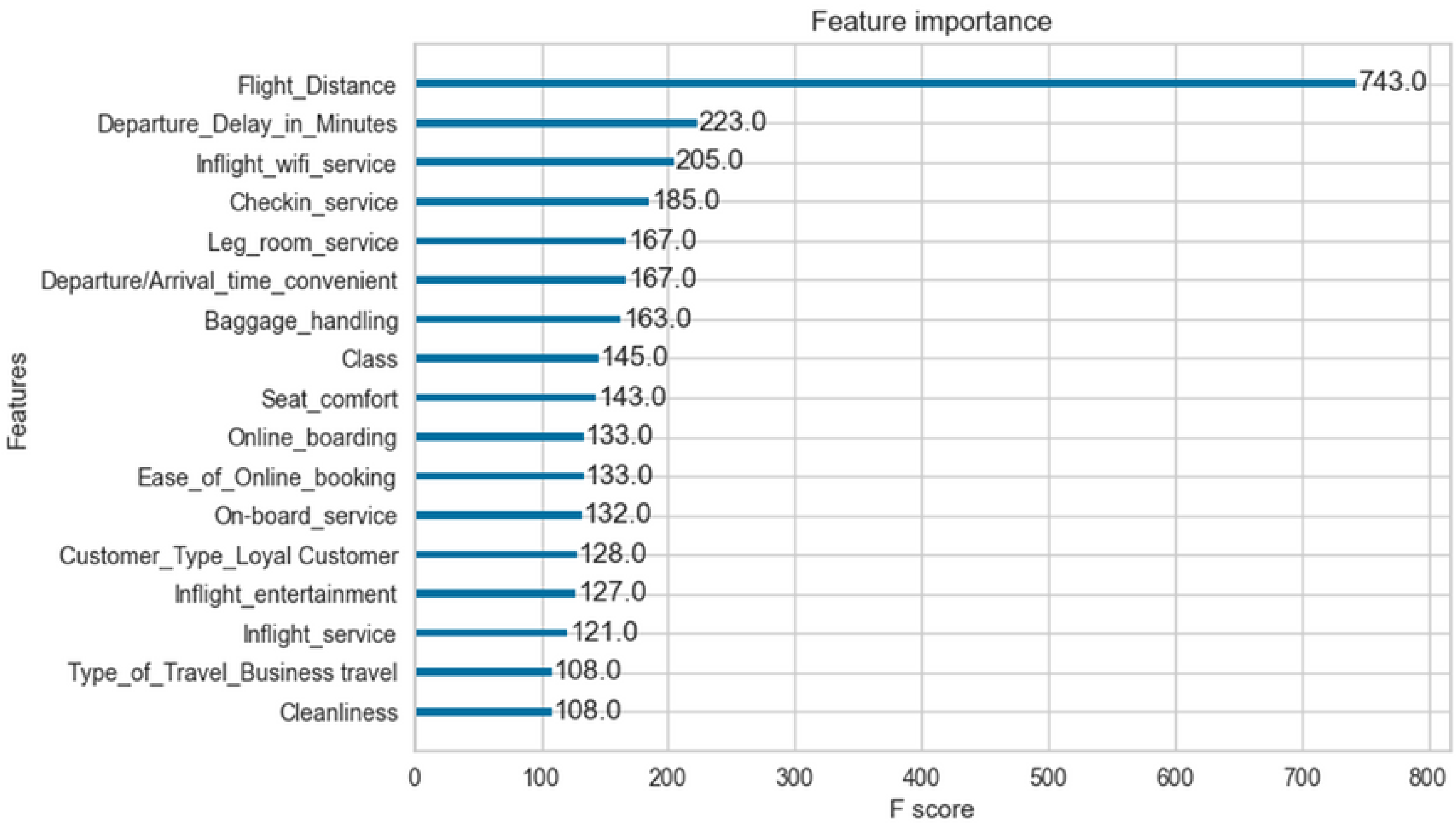
	precision	recall	f1-score	support
DummyClassifier	0.535878	1.000000	0.697813	2345
LogisticRegression	0.933783	0.944136	0.938931	2345
KNN	0.968830	0.967591	0.968210	2345
Naive Bayesian	0.940160	0.904478	0.921973	2345
Decision Tree	0.973493	0.971002	0.972246	2345
Random Forest	0.981489	0.972281	0.976864	2345
LightGBM	0.981148	0.976546	0.978842	2345
SVM	0.961961	0.970576	0.966249	2345
AdaBoost	0.934379	0.965458	0.949664	2345

LightGBM is the best supervised technique

Final Insights

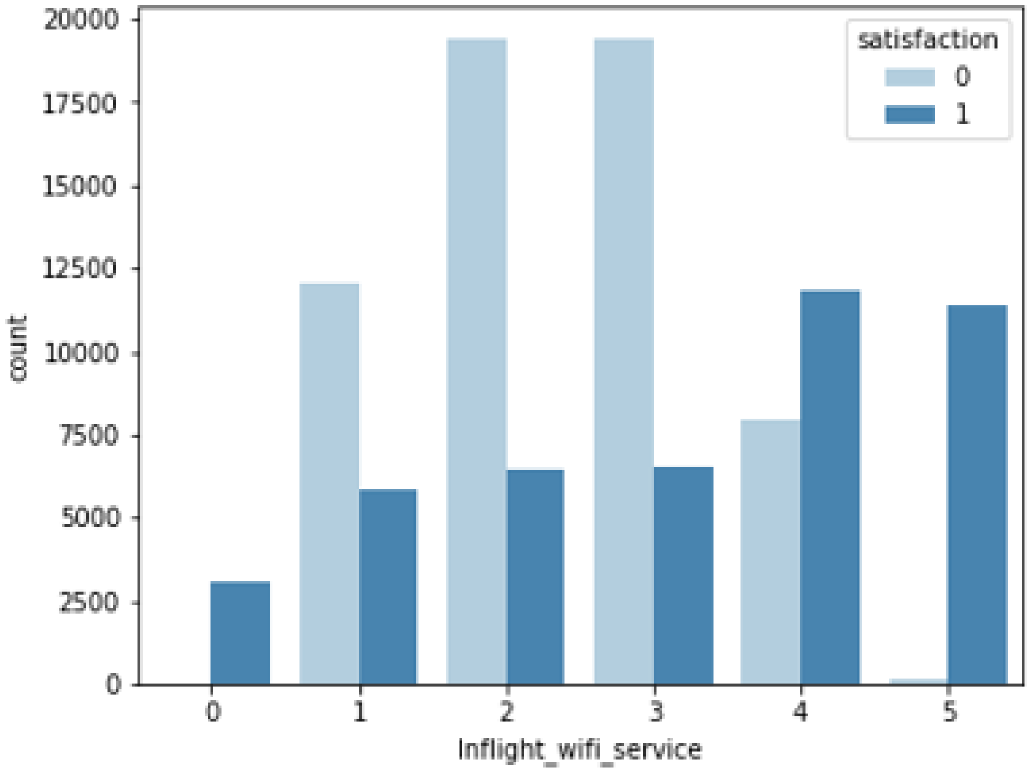
Feature Importance

We observe the feature importance of the most accurate model (LightGBM) and conclude that Flight_Distance, Departure_Delay_in_Minutes and Inflight_wifi_service are among the most important features to accurately predict the customer satisfaction.



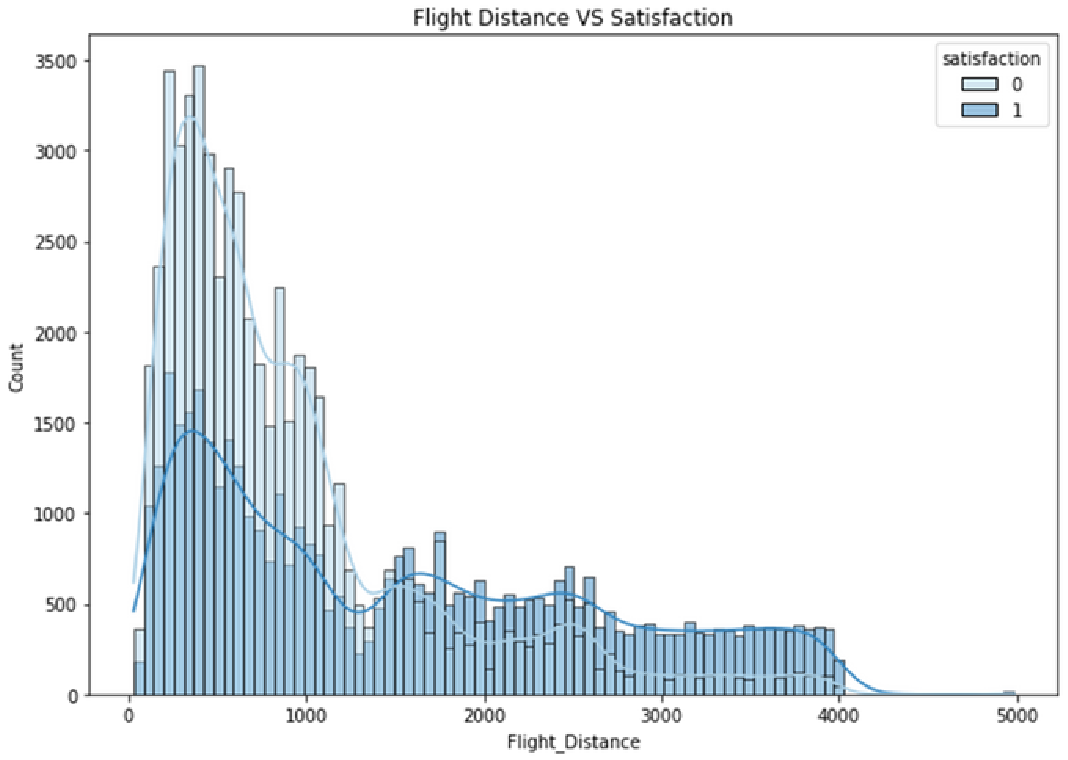
In-Flight Wifi Service

The customer satisfaction is increasing and dissatisfaction is decreasing as the WIFI service rating increase.



Flight Distance

Lower distance flights are where the most focus on increasing satisfaction should be made as these are where the most complaints are being generated.



EDA Results

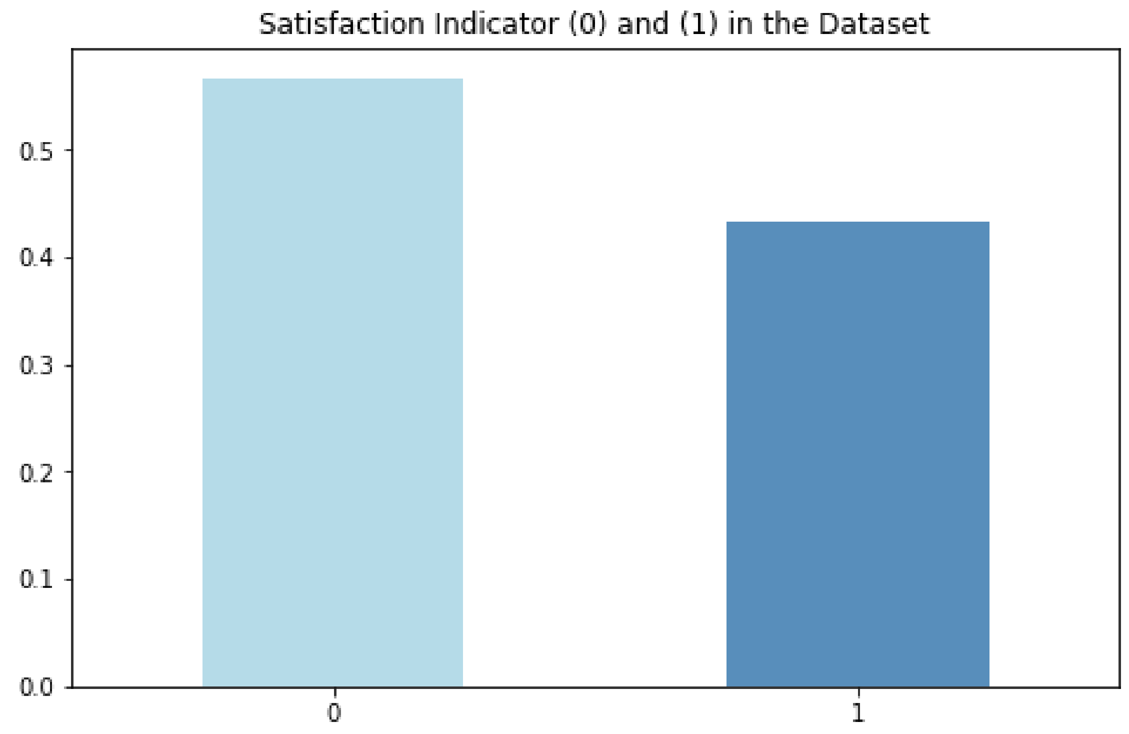
Dataset Overview

- Target Column - satisfaction (*satisfied: 1, neutral or dissatisfied: 0*)
- Passengers' ratings for flight services such as check-in, in-flight wifi, food & drinks, and seat comfort.
- Passengers' attributes such as age, gender, loyal/disloyal customer, business/personal travel.

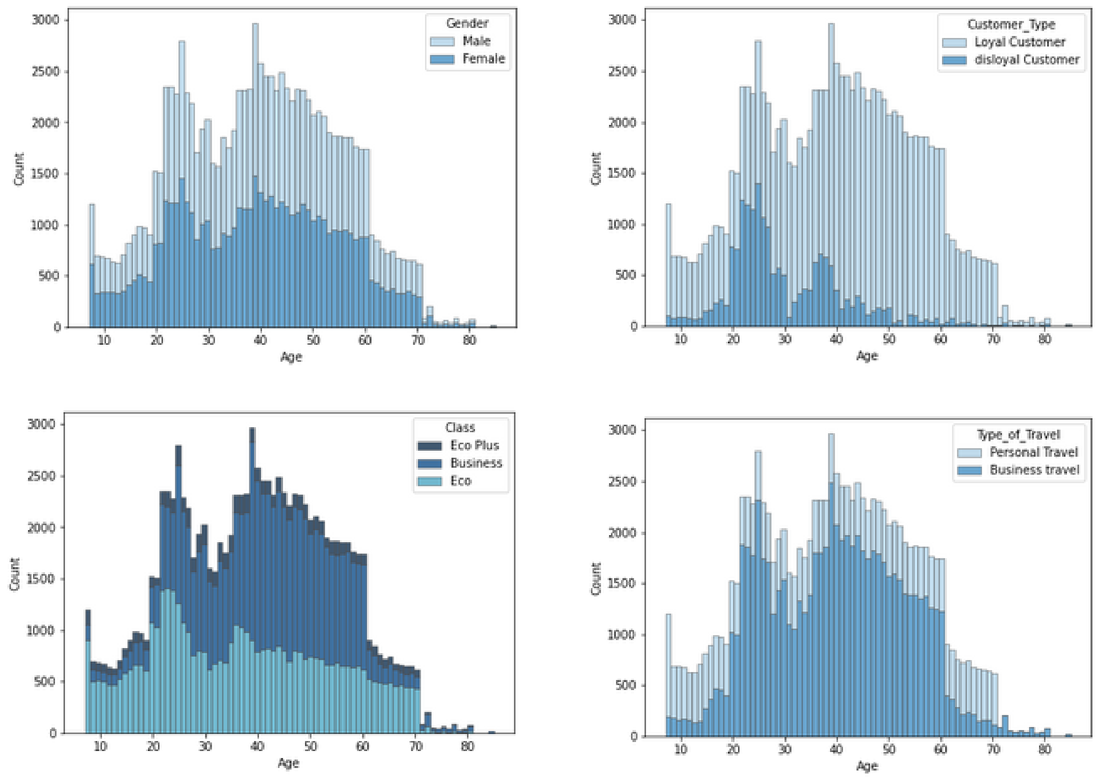
Data Preparation

- Treating categorical variables
  - Ordinal encoding for travel class
  - One-Hot Encoding - gender, customer type, type of travel
- Removed outliers using Isolation Forest
- Treated missing age values using SimpleImputer with median strategy

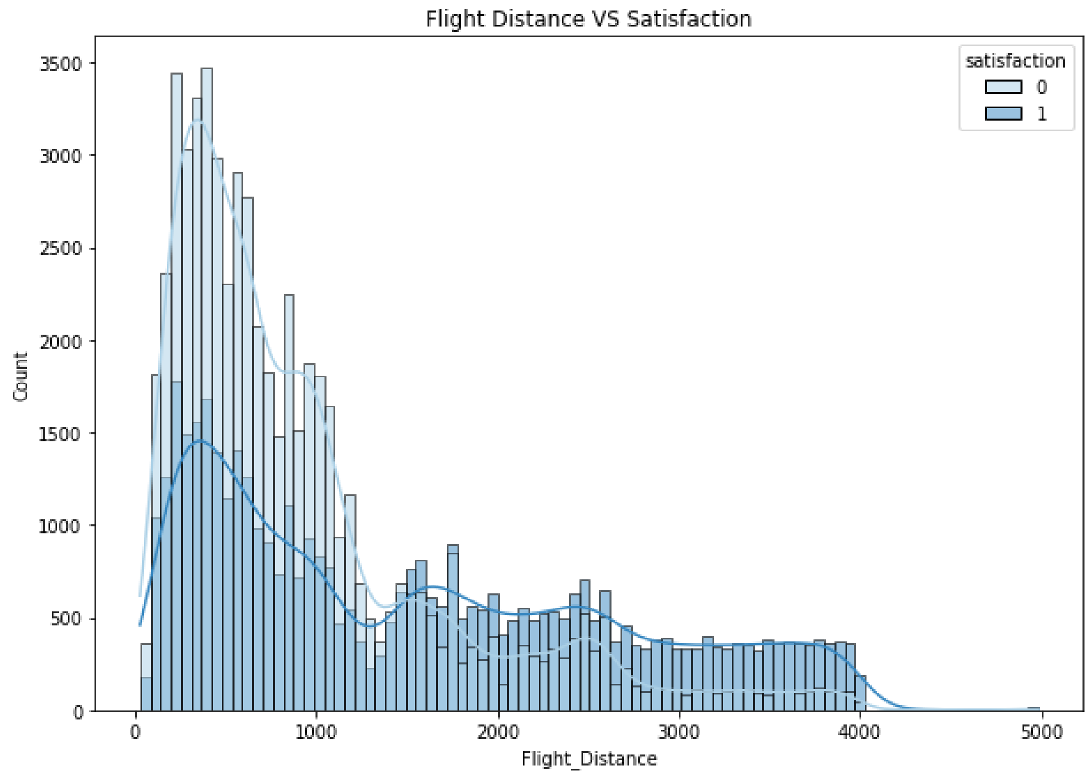
EDA - Visualizations



Distribution of Target Variable



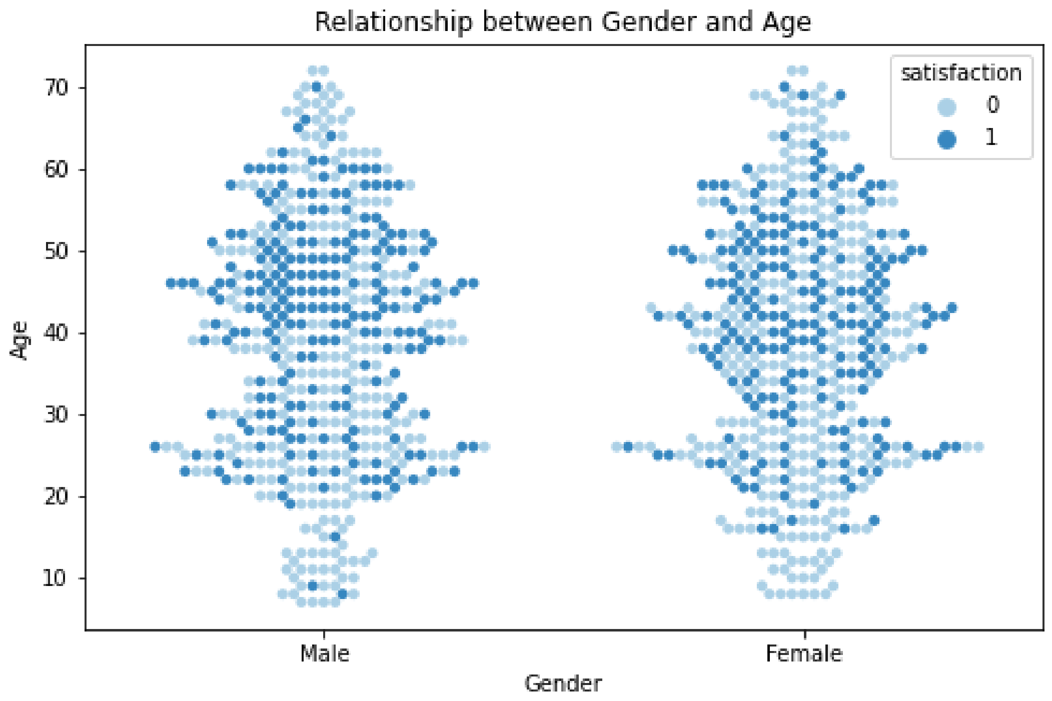
Relationship among categorical variables



Flight Distance vs Satisfaction

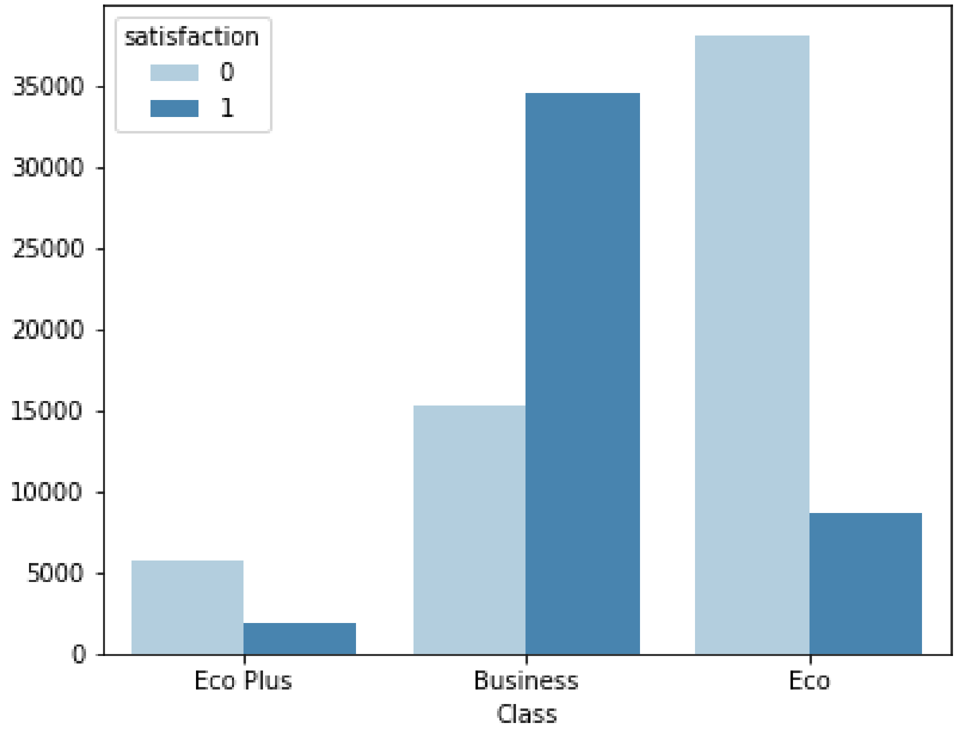
Data Insights

- Average satisfaction can vary based on age and gender
- Females are on average less satisfied than males with flying
- Idem for the oldest age group compared to others



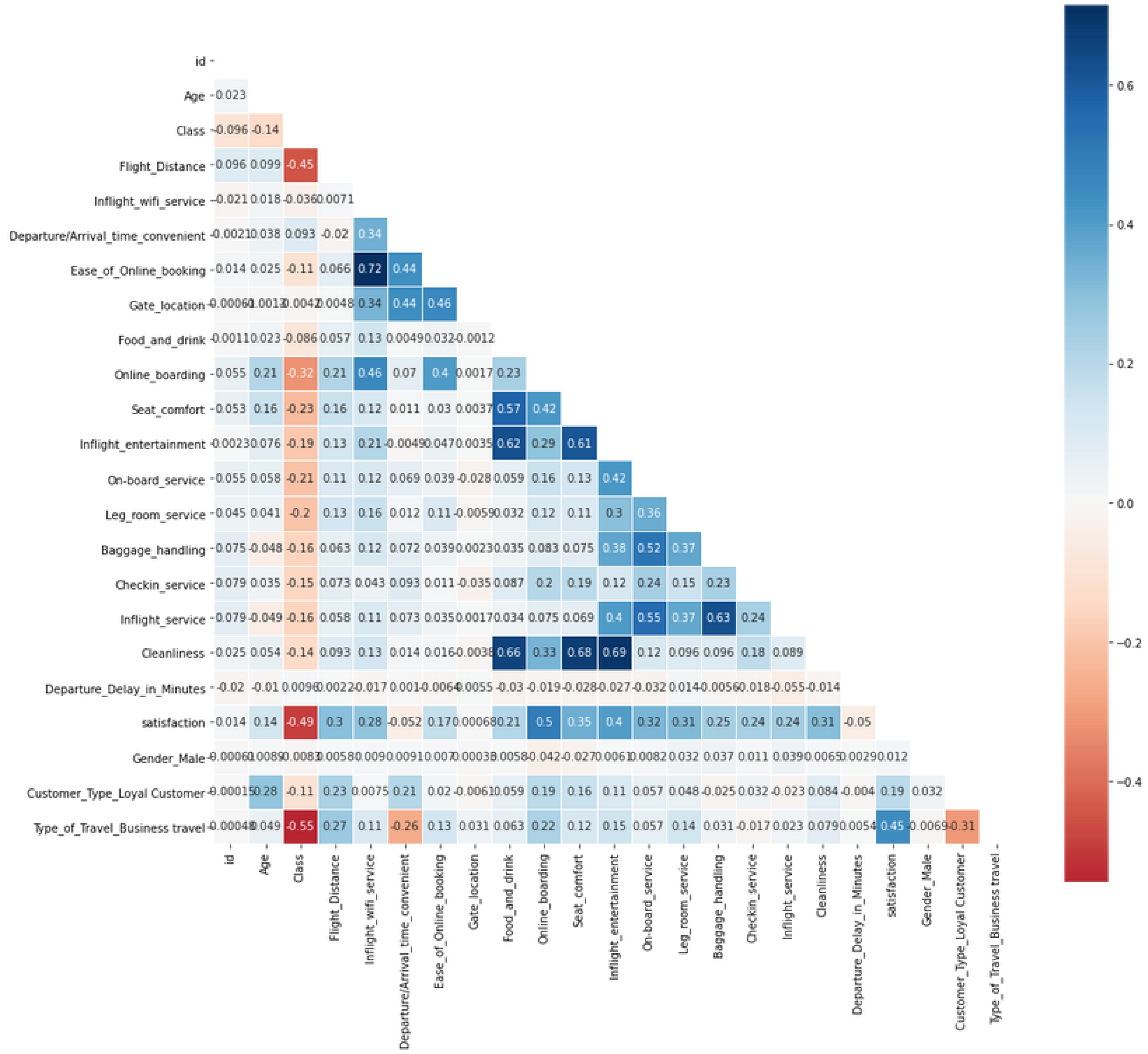
Females had an average satisfaction rate of 42.74% compared to 43.95% for males. Individuals aged between 40 and 60 years old are most satisfied.

- Idem for flying class and purpose of travel.
- Passenger satisfaction rates significantly increase when compared with the class they are flying in.
- One possible factor is that for business travel, flights are often subsidized by the company who chooses business class (hence the name) and thus passengers are more satisfied as they are essentially flying for free.



Business Class travelers were satisfied 69.43% of the time.
Passengers who flew for business similarly had a 58.26% satisfaction rate.

Correlation Matrix



- Negative correlation between distance and class
- Cleanliness being positively correlated with food, inflight entertainment and seat comfort
- Idem for inflight handling and baggage service
- Loyal customers being more likely to travel for business and upgrade their class
- Users who book online are more likely to use in flight Wi-fi and be dissatisfied by it