
Hotel Cancellations Productionisation

McGill-MMA-EnterpriseAnalytics/[hotel_cancellation_ML2](#)

Alisa Liu	18ZL107
Chiara Lu	LuChiara
Hao Duong	haohaoduong
Jaya Chaturvedi	Jaya2404
Keani Schuller	keanischuller
Reo Paul Jackson	reojackson31



TABLE OF CONTENTS

•01•

HYPER-PARAMETER TUNING

All tuning done on previous project's model

•02•

DOCKER DEPLOYMENT

Using FastAPI & Gradio deployed on Docker

•03•

PRODUCTION ARCHITECTURE

Cloud architecture using Databricks

•04•

DRIFT, FAIRNESS & EXPLAINABILITY

Evaluation using adversarial & KSdrift, FairML, and LIME

TEAM & ROLES



Jaya Chaturvedi
Data Engineer



Keani Schuller
Business Analyst



Chiara Lu
Data Scientist / PM



Reo Paul Jackson
ML Engineer



Alisa Liu
Data Scientist



Hao Duong
Business Analyst

ABOUT HOTEL CANCELLATIONS

HOTEL CANCELLATIONS ML1 SUMMARY



CONTEXT

Managing hotel cancellations is crucial for maintaining revenue and enhancing guest experiences



MODEL

Classification model experimentation to predict the likelihood of cancellations



INTERPRETATION

Random Forest as final model and feature importance to get insights

BUSINESS CASE

ACCURACY

Provide hoteliers with the ability to forecast the likelihood of a reservation being canceled at the time of booking using the highest accuracy model

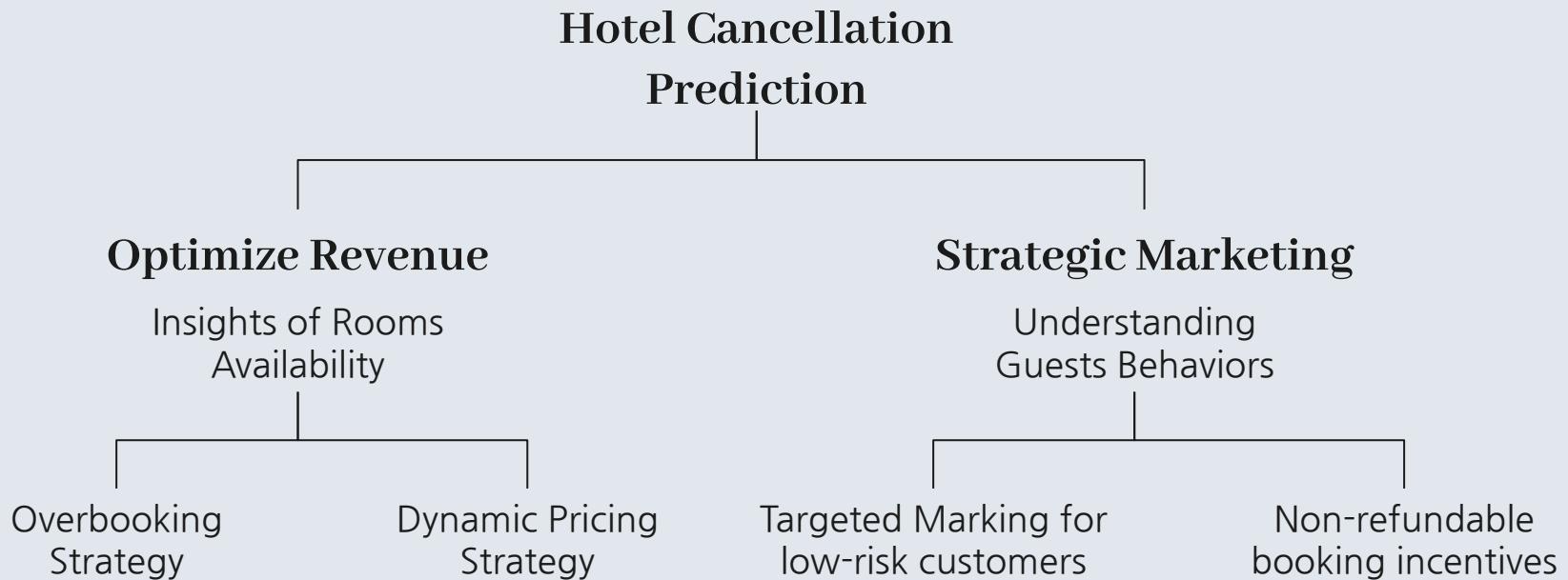
LONGEVITY

Audit the model to ensure it is easily interpreted, fair, and has longevity

DEPLOYMENT

Deployment is both local, using Docker containers, and on the cloud via Databricks, giving flexibility and scalability in operations

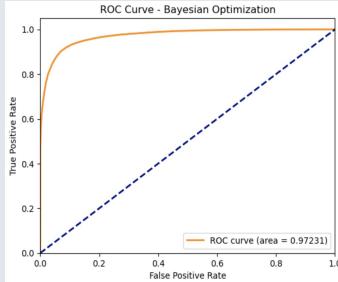
VALUE PROPOSITION



HYPERPARAMETER TUNING

HYPERPARAMETER TUNING

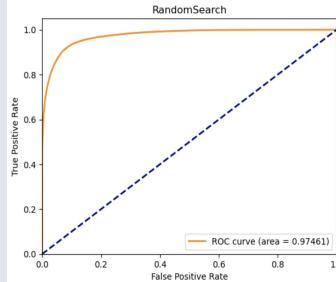
Bayesian Optimization



Best Params:

- max_depth: 30
- max_features: 1
- min_samples_leaf: 1
- min_samples_split: 2
- n_estimators: 201

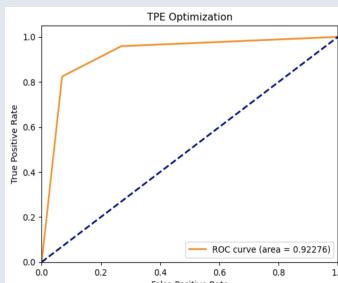
RandomizedSearchCV



Best Params:

- max_depth: None
- max_features: 'sqrt'
- min_samples_leaf: 1
- min_samples_split: 5
- n_estimators: 100

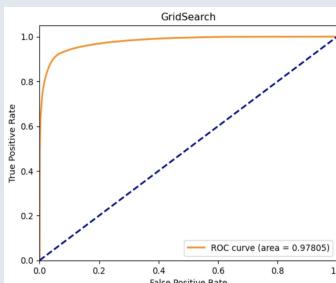
TPE Optimization (HyperOpt)



Best Params:

- max_depth: None
- max_features: 0.236
- min_samples_leaf: 0
- min_samples_split: 0
- n_estimators: 2

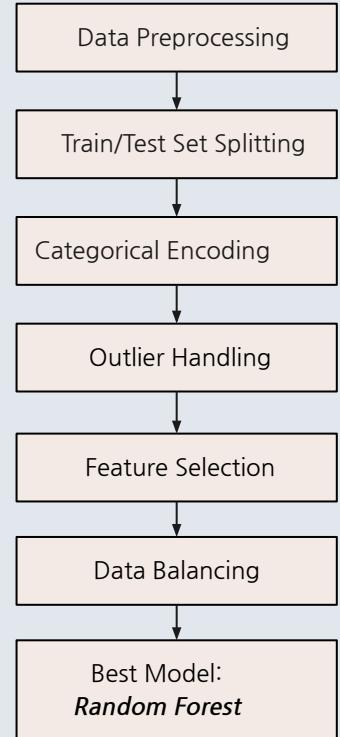
GridSearchCV



Best Params:

- max_features: None
- min_samples_leaf: 1
- n_estimators: 200

Part I Base Model



AutoML

Key Benefits

1. Import train, test, and validation datasets
2. Library installations



- Light
- Neural Network
- Multifactor Dimensionality Reduction

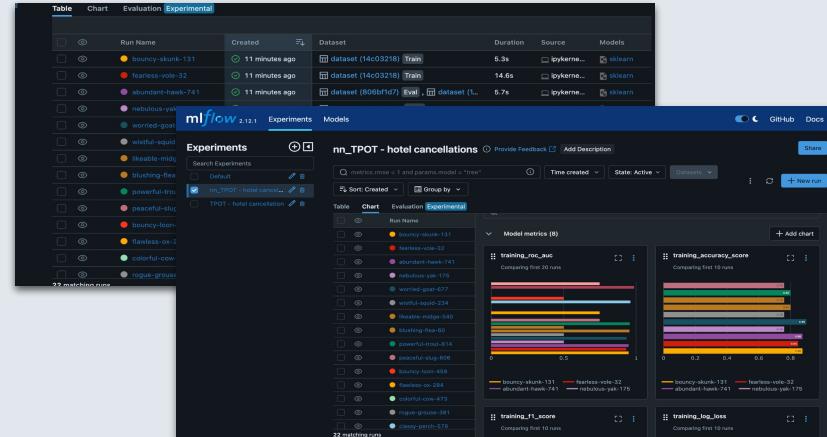
3. Initiate Model
4. Train
5. Export Pipeline

Efficient

Accessible

Reproducible

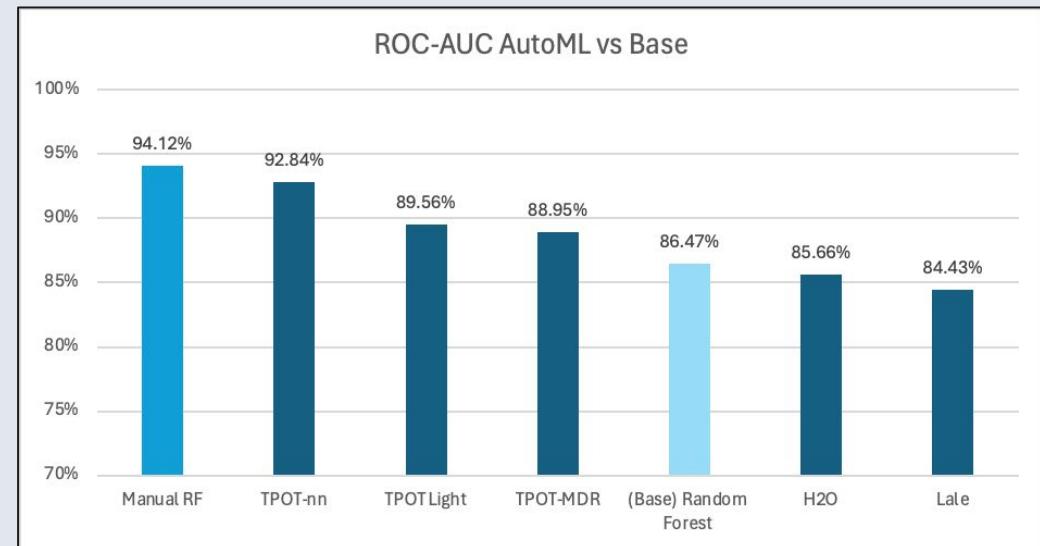
mlflow™



MODELING RESULTS

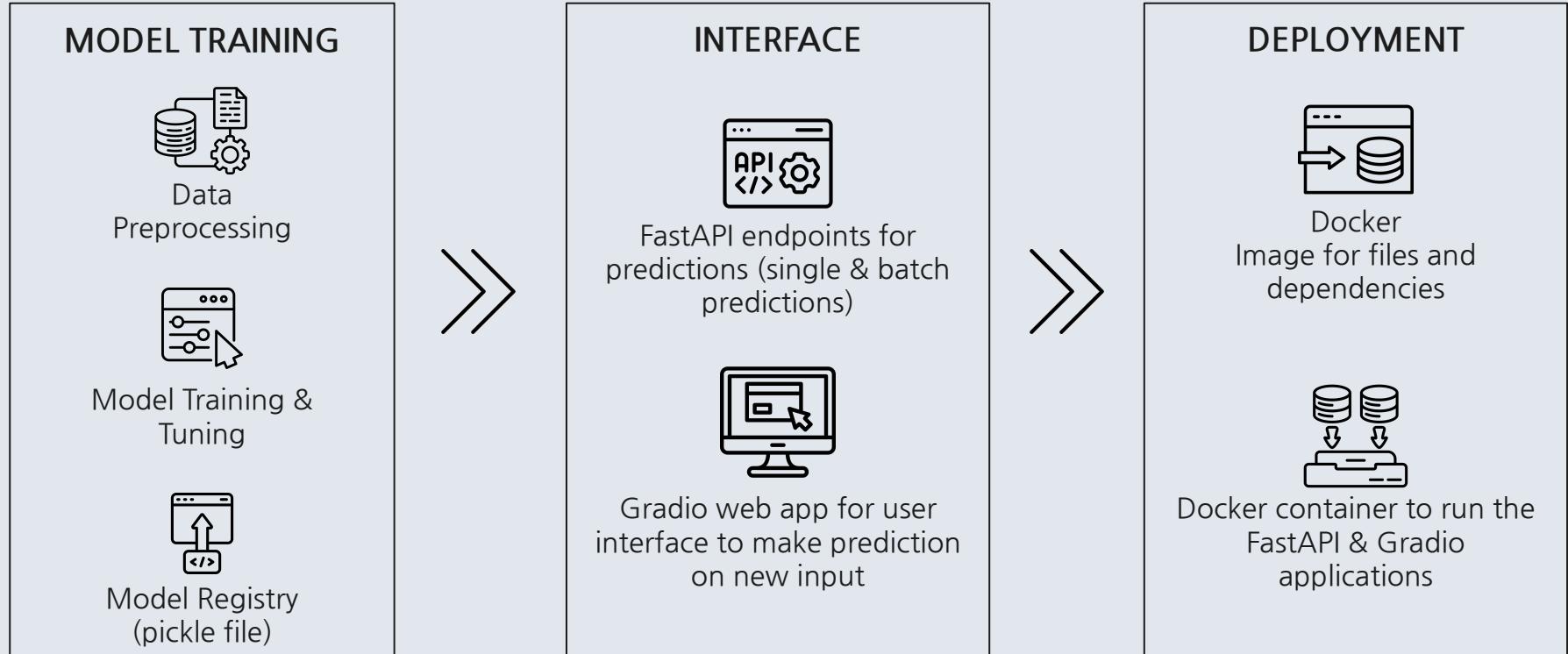
- ROC-AUC was used as the comparison metric because it discriminates how well a model is able to perform a binary classification
- The best pipeline from the AutoML approaches were the TPOT variations
- However, pipelines from AutoML did not prove to be better than the manually fine-tuned pipeline

Model	ROC-AUC
Manual RF	94.12%
TPOT-nn	92.84%
TPOT Light	89.56%
TPOT-MDR	88.95%
(Base) Random Forest	86.47%
H2O	85.66%
Lale	84.43%



DOCKER DEPLOYMENT

FastAPI & Gradio



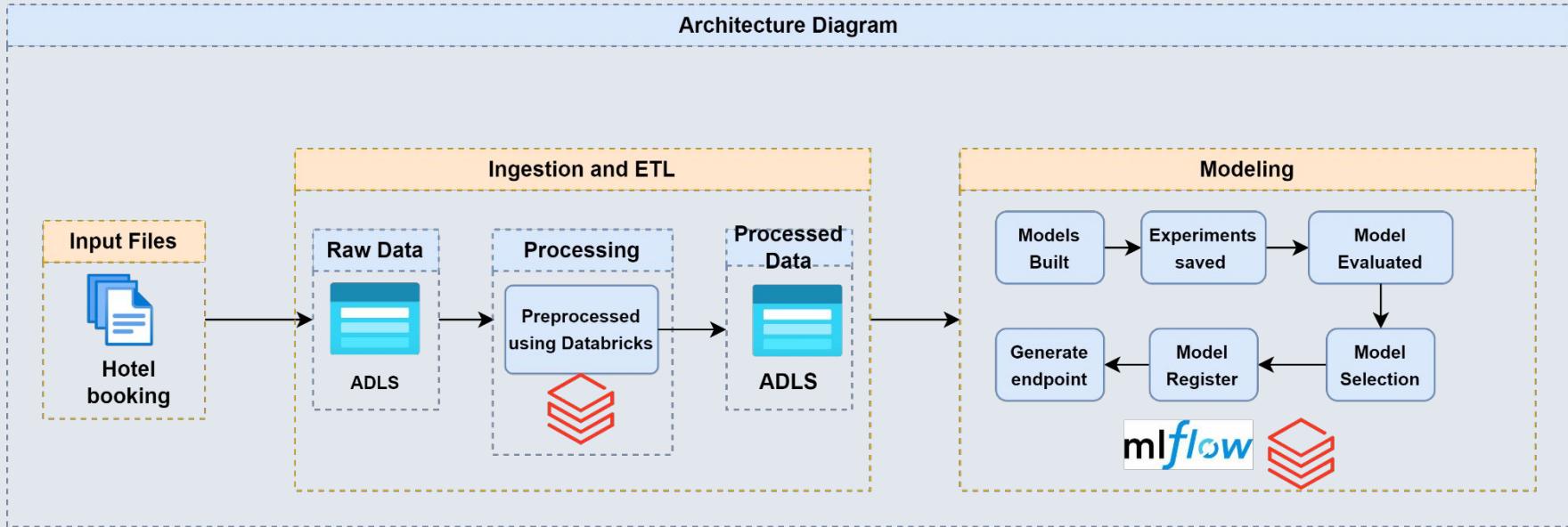


DEMO

ARCHITECTURE

USING DATABRICKS

Architecture Diagram



USING DATABRICKS

<input type="checkbox"/>	RandomForest_Champion	<input checked="" type="checkbox"/> 1 day ago	11.0min	best_mo...	RandomForest_Champion ...	-	0.978017118...	0.870560643...	0.941280977...
<input type="checkbox"/>	fearless-finch-111	<input checked="" type="checkbox"/> 1 day ago	11.3min	best_mo...	sklearn	0.978017118...	-	-	-
<input type="checkbox"/>	adorable-grub-283	<input checked="" type="checkbox"/> 1 day ago	7.6min	best_mo...	-	0.881166351...	-	-	-
<input type="checkbox"/>	TPOT_model	<input checked="" type="checkbox"/> 1 day ago	5.0min	best_mo...	-	0.919666934...	-	-	-
<input type="checkbox"/>	RandomForest_BestParam...	<input checked="" type="checkbox"/> 1 day ago	12.2min	best_mo...	RandomForest_BestParam...	0.978017118...	-	-	-

Serving endpoints >

BestParameters

Serving endpoint state: Ready
Inference table: Not enabled

Created by: jaya.chaturvedi@mail.mcgill.ca
Route optimization: Not enabled

URL: <https://adb-4634864076059286.azuredatabricks.net/serving-endpoints/BestParameters/invocations>

Tags:

Active configuration

Entity	Version	Name	State	Compute	Traffic (%)
RandomForest_BestParameters	Version 1	RandomForest_BestParameters-1	Ready	CPU, Small 0-4 concurrency (0-4 DBU)	100

USING DATABRICKS

The screenshot shows the Databricks Query endpoint interface. On the left, there's a sidebar with 'Serving endpoints' and 'BestParameters' sections, and tabs for 'Metrics' and 'Events'. The main area has tabs for 'Browser', 'Curl', 'Python', and 'SQL', with 'Browser' selected. The 'Request' section contains a JSON object representing feature columns:

```
{  
  "dataframe_split": {  
    "columns": [  
      "lead_time",  
      "arrival_date_week_number",  
      "arrival_date_day_of_month",  
      "stays_in_weekend_nights",  
      "stays_in_week_nights",  
      "adults",  
      "children",  
      "babies",  
      "is_repeated_guest",  
      "previous_cancellations",  
      "previous_bookings_not_canceled",  
      "booking_changes",  
      "days_in_waiting_list",  
      "adr",  
      "required_car_parking_spaces",  
      "total_of_special_requests",  
    ]  
  }  
}
```

The 'Response' section shows the JSON output from the 'RandomForest_BestParameters-1' endpoint:

```
{  
  "predictions": [  
    1  
  ]  
}
```

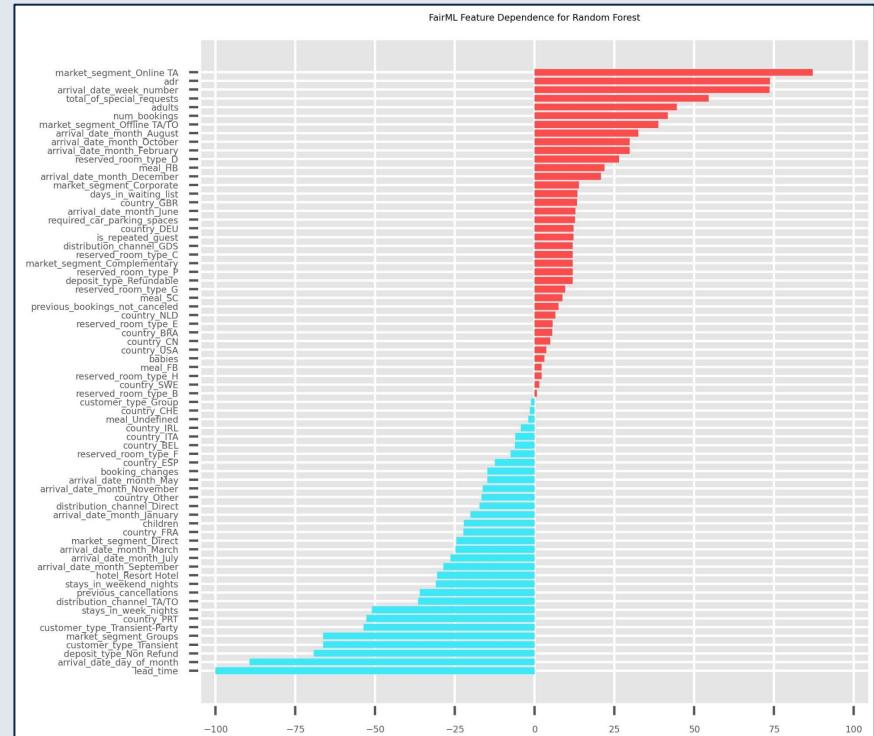
At the bottom, there are buttons for 'Send request' and 'Reset example'.



FAIRNESS & EXPLAINABILITY

MODEL FAIRNESS

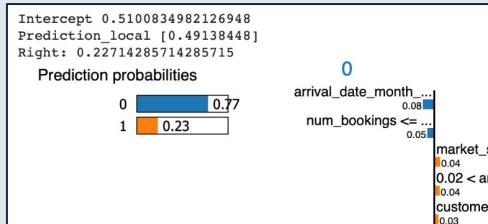
- Used FairML to evaluate the fairness of our final model
- market_segment_Online TA (Online Travel Agent) has the highest positive influence
- Reservations made through online travel agents significantly increase the likelihood that the model predicts a cancellation
- Conversely, lead_time has the highest negative influence
- Reservations made well in advance decrease the likelihood it predicts a cancellation



MODEL EXPLAINABILITY WITH LIME

PREDICTION

Model predicts that there's a 77% chance that the reservation will not be cancelled and a 23% chance that it will



POSITIVE INFLUENCE

Arrival month's value of 0.59 suggests that the month of the arrival date contributes towards a higher chance of cancellation

NEGATIVE INFLUENCE

Variables like children, babies and is_repeated_guest all have negative values meaning they contribute to a lesser likelihood of cancellation

Feature	Value
lead_time	-0.50
arrival_date_week_number	1.16
arrival_date_day_of_month	0.59
stays_in_weekend_nights	-0.94
stays_in_week_nights	-0.25
adults	0.25
children	-0.25
babies	-0.07
is_repeated_guest	-0.17
previous_cancellations	-0.10
previous_bookings_not_canceled	-0.10
booking_changes	-0.33
days_in_waiting_list	-0.13
adr	-0.14

MODEL DRIFT

A wide-angle photograph of a coastal landscape. In the foreground, dark, jagged rocks jut out from the ocean. To the right, a paved walkway leads towards a building with a red-tiled roof, partially obscured by greenery and a tall palm tree. The ocean stretches to the horizon under a hazy sky.

MODEL DRIFT

Adversarial Drift

Adversarial Drift Test Setup

```
def detect_adversarial_examples(clf_rf_best, X, threshold=0.5):

    # Get prediction probabilities for each class
    probs = clf_rf_best.predict_proba(X)
    # Get the maximum prediction probability for each sample
    max_probs = np.max(probs, axis=1)
    # Flag as potential adversarial if max probability is below the threshold
    return max_probs < threshold

# Example usage
adversarial_flags = detect_adversarial_examples(clf_rf_best, X_test)

print("Potential adversarial detected:", adversarial_flags)
```

Potential adversarial detected: [False False False ... False False False]

There is no adversarial drift detected in the model

KSDrift

KSDrift Setup

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder

# Train a Random Forest classifier
clf = RandomForestClassifier()
clf.fit(X_ref, y_ref)

# Generate predictions for the reference and new dataset
y_ref_pred = clf.predict_proba(X_ref)
y_new_pred = clf.predict_proba(X_new)
|
ohe = OneHotEncoder(sparse=False, categories='auto', handle_unknown='ignore')
y_ref_ohe = ohe.fit_transform(y_ref_pred.reshape(-1, 1))
y_new_ohe = ohe.transform(y_new_pred.reshape(-1, 1))

# Initialize KSDrift detector
cd = KSDrift(p_val=0.05, x_ref=y_ref_ohe)

# Check for drift in the new predictions
drift_results = cd.predict(y_new_ohe)

# Output
print(f'Drift detected: {drift_results["data"]["is_drift"]}')
```

Drift detected: 0



THANK YOU!

Q&A

APPENDIX

EXPORT TPOT-NN

```
# export best nn_tpot
nn_tpot.export('nn_tpot_pipeline.py')

✓ 0.0s
```



```
❸ nn_tpot_pipeline.py > ...
1 import numpy as np
2 import pandas as pd
3 from sklearn.model_selection import train_test_split
4 from sklearn.pipeline import make_pipeline, make_union
5 from sklearn.tree import DecisionTreeClassifier
6 from tpot.builtins import StackingEstimator
7 from xgboost import XGBClassifier
8
9 # NOTE: Make sure that the outcome column is labeled 'target' in the data file
10 tpot_data = pd.read_csv('PATH/TO/DATA/FILE', sep='COLUMN_SEPARATOR', dtype=np.float64)
11 features = tpot_data.drop('target', axis=1)
12 training_features, testing_features, training_target, testing_target = \
13     train_test_split(features, tpot_data['target'], random_state=None)
14
15 # Average CV score on the training set was: 0.9310294569353934
16 exported_pipeline = make_pipeline(
17     StackingEstimator(estimator=DecisionTreeClassifier(criterion="gini", max_depth=4, min_samples_leaf=4, min_samples_split=9)),
18     XGBClassifier(learning_rate=0.5, max_depth=5, min_child_weight=11, n_estimators=100, n_jobs=1, subsample=0.6000000000000001, verbosity=0)
19 )
20
21 exported_pipeline.fit(training_features, training_target)
22 results = exported_pipeline.predict(testing_features)
```

MLOps BEST PRACTICES

- Project Management using Kanban
- Keeping experiment logs on MLflow
- Working in squads to attack a major problem in small steps
- Use GitLens to view changes on GitHub branches
- Save failed tests

The screenshot shows a Jira board titled "Productionizing Hotel Cancellation SaaS" with three columns: "Backlog", "Team capacity", "Current iteration", "Roadmap", "My items", "View 6", and "New view". The "Current iteration" column is selected. It contains three main sections: "Todo", "In Progress", and "Done".

- Todo:** 5 / 10 Estimate: 0. Items include: "This item hasn't been started", "Business Case", "Model deployment: Synthetic streaming data", "Pass best models: pkl for Docker, ipynb for Databricks", "Architecture diagram", and "Presentation slide deck".
- In Progress:** 4 / 8 Estimate: 0. Items include: "Streamline all steps in part 1 + hyperparameter tuning", "Model Deployment: Database setup", "Model Explainability (ELI5, SHAP/LIME)", and "Evaluate Fairness/Bias in predictions".
- Done:** 6 / 8 Estimate: 0. Items include: "Model pipeline for data preprocessing & predictions", "Model deployment using FastAPI", "Model deployment using Docker container", "Model deployment using Cloud setup", "AutoML - explore 2-3 approaches", and "Model Drift".



A screenshot of the GitLens extension for GitHub. The interface shows a detailed commit history for a branch named "alisa-tuning". The commits are listed in chronological order, showing author names like Alisa Liu, Reo Paul Jackson, and Keani, along with their commit messages and timestamps. The commits include actions such as "Add files via upload", "Create Image", "added MLflow", "add files for docker", "add changes for fastapi", "add model training pipeline", "Delete SHAPBLIME/Test", "Add files via upload", "Merge branch 'main'", "Add files via upload", "create branch", "Create Test", "Delete FairML/FairML_Test", "Delete Mounting ADLS Container.ipynb", and "Add files via upload". The commits also show file changes and commit times.

A screenshot of the MLflow UI. The top navigation bar includes "mlflow", "Dashboard", "Experiments", "Metrics", "Sources", and "Models". The main area shows a list of experiments under "Experiments" and a detailed view of the "mr_TPOT - hotel cancellations" experiment. The experiment details include "Dataset", "Metrics", "Sources", and "Modules". Below this, there are tabs for "Table", "Evaluation", and "Experiments". The "Experiments" tab shows a list of runs with metrics like "dataset_1", "dataset_2", and "dataset_3". The "Metrics" tab displays various performance metrics for different runs, and the "Sources" tab shows the data sources used. The bottom right corner shows a preview of the "mr_TPOT" dashboard.

DOCKER IMAGE (FOR FASTAPI)

```
1 FROM python:3.11
2
3 WORKDIR /code
4
5 COPY ./requirements.txt /code/requirements.txt
6
7 RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt
8
9 #
10 COPY ./preprocessing.py /code/
11 COPY ./bestModel_hpo.pkl /code/
12 COPY ./bestModel_tpot.pkl /code/
13 COPY ./fastapi_app.py /code/
14
15 #
16 CMD ["uvicorn", "fastapi_app:app", "--host", "0.0.0.0", "--port", "8000"]
```