CAPSTONE PROJECT MODULE 3 & 4 - MACHINE LEARNING & CLOUD COMPUTING

Bike Sharing Supply Optimization with Machine Learning Prediction

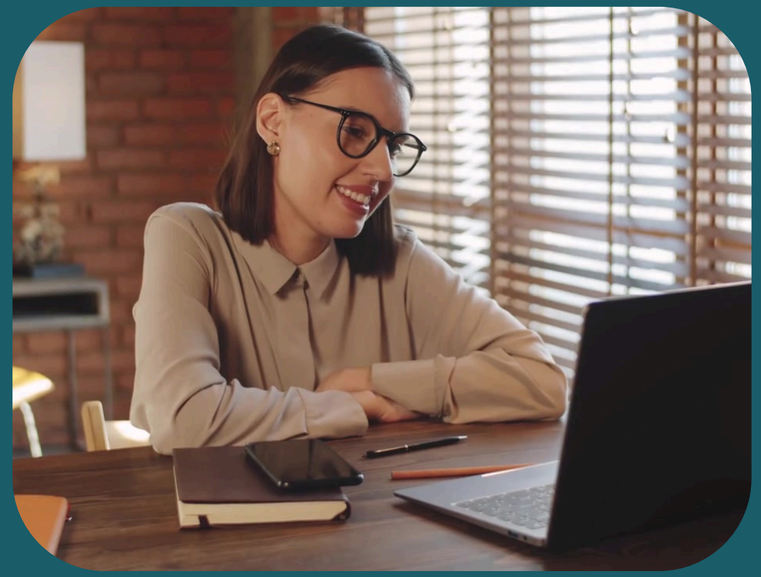
---

Safira Fabilia - DTI DS 0106 001


Safira Fabilia
DTI DS Student

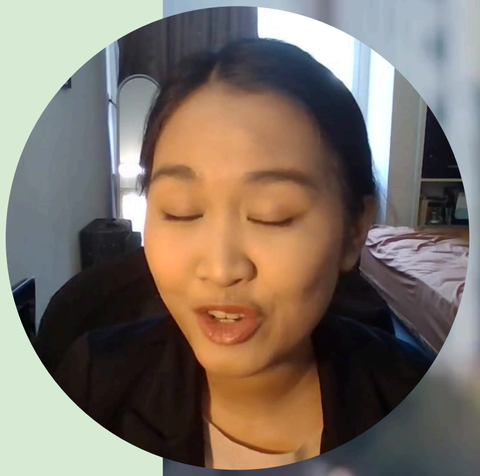
Three main contents of this video

1. Business Case Overview
2. Machine Learning Model Deployment
3. Final Thoughts



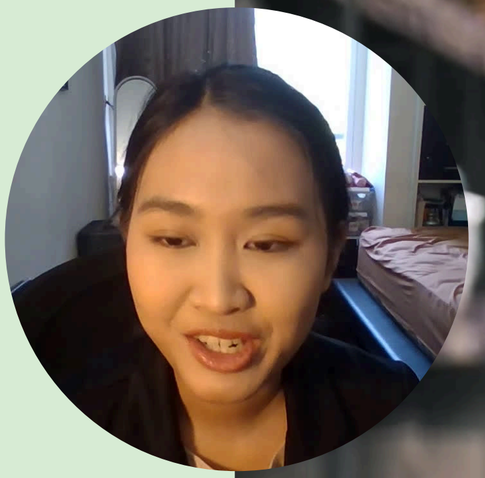
Part 1

Business Case Overview

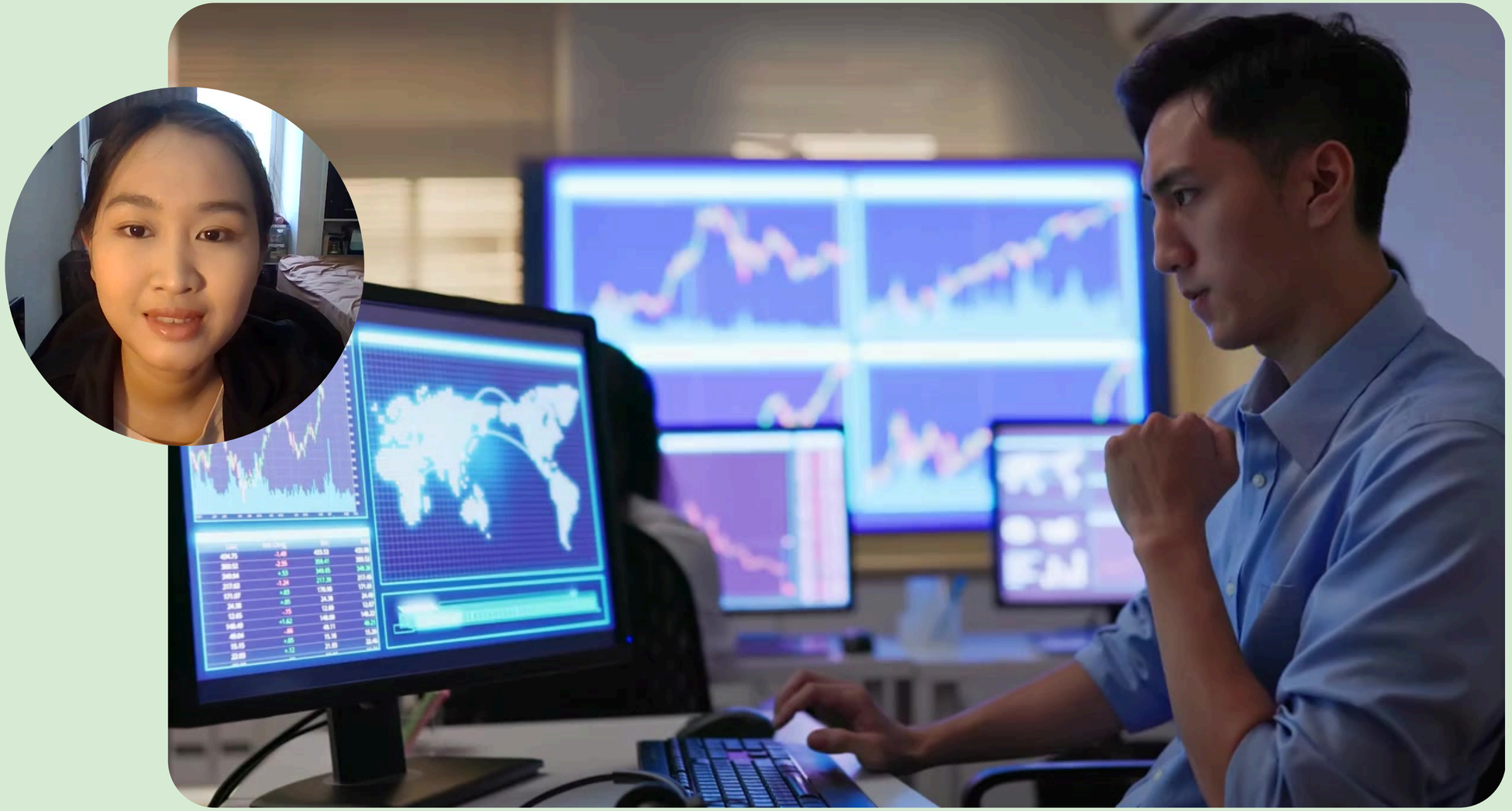


Bike sharing is a transportation system where bicycles are made available for shared use to individuals on a short-term basis

A fictional business scenario that illustrates how a bike sharing company use a machine learning approach for demand prediction

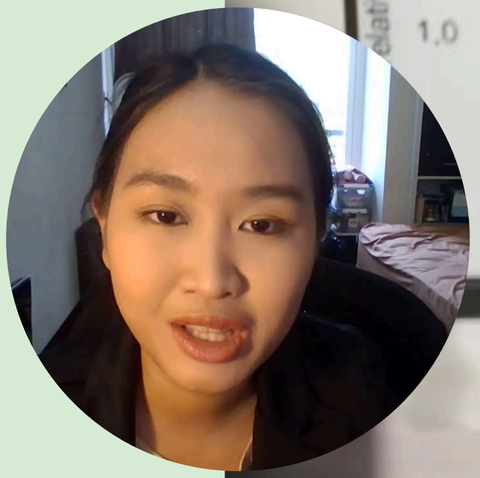


You are the owner of a bike-sharing company that has been operational since 2010.



Additionally, this presents an opportunity to revamp your management strategies by integrating advanced technologies and data-driven approaches.

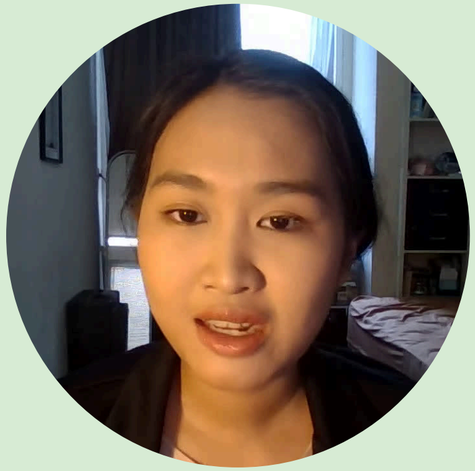
Predicting demand with Machine Learning Regression, ensuring that bikes are available whenever needed to improve customer satisfaction.



Thus, you hired a data scientist who is tasked with forecasting the number of bike rentals to ensure sufficient bikes are available to meet demand.

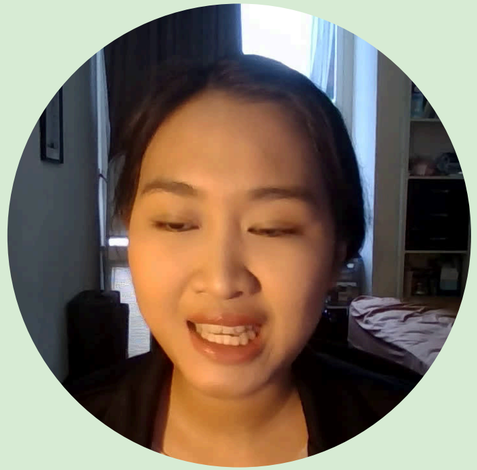
OBJECTIVES

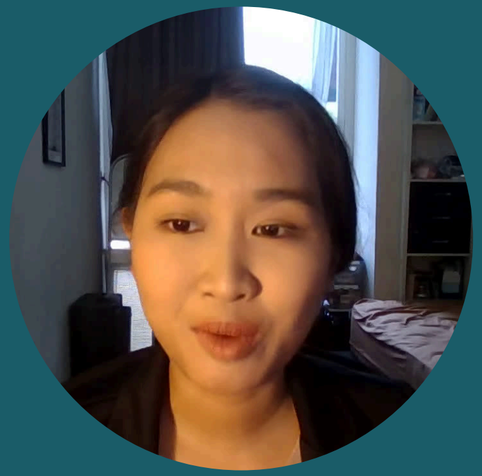
1. How can we ensure that our predictive model is robust enough for practical use?
2. Based on our predictive model, what is the estimated number of bikes needed to meet demand on specific days under certain conditions?
3. How do we know that this prediction is beneficial to the company?
4. What factors influence bike rental demand based on the dataset?
5. What actionable insights and recommendations can be derived to ensure efficient bike management?



GOALS

1. Ensure that the predictive model achieves good evaluation metric scores, using MSE, MAE, and R2 as the goals.
2. Predict the number of bikes needed using the machine learning model.
3. Evaluate how the predicted demand translates into revenue.
4. Identify and analyze the factors that influence bike rental demand.
5. Recommend insights to make sure that bikes are managed efficiently.



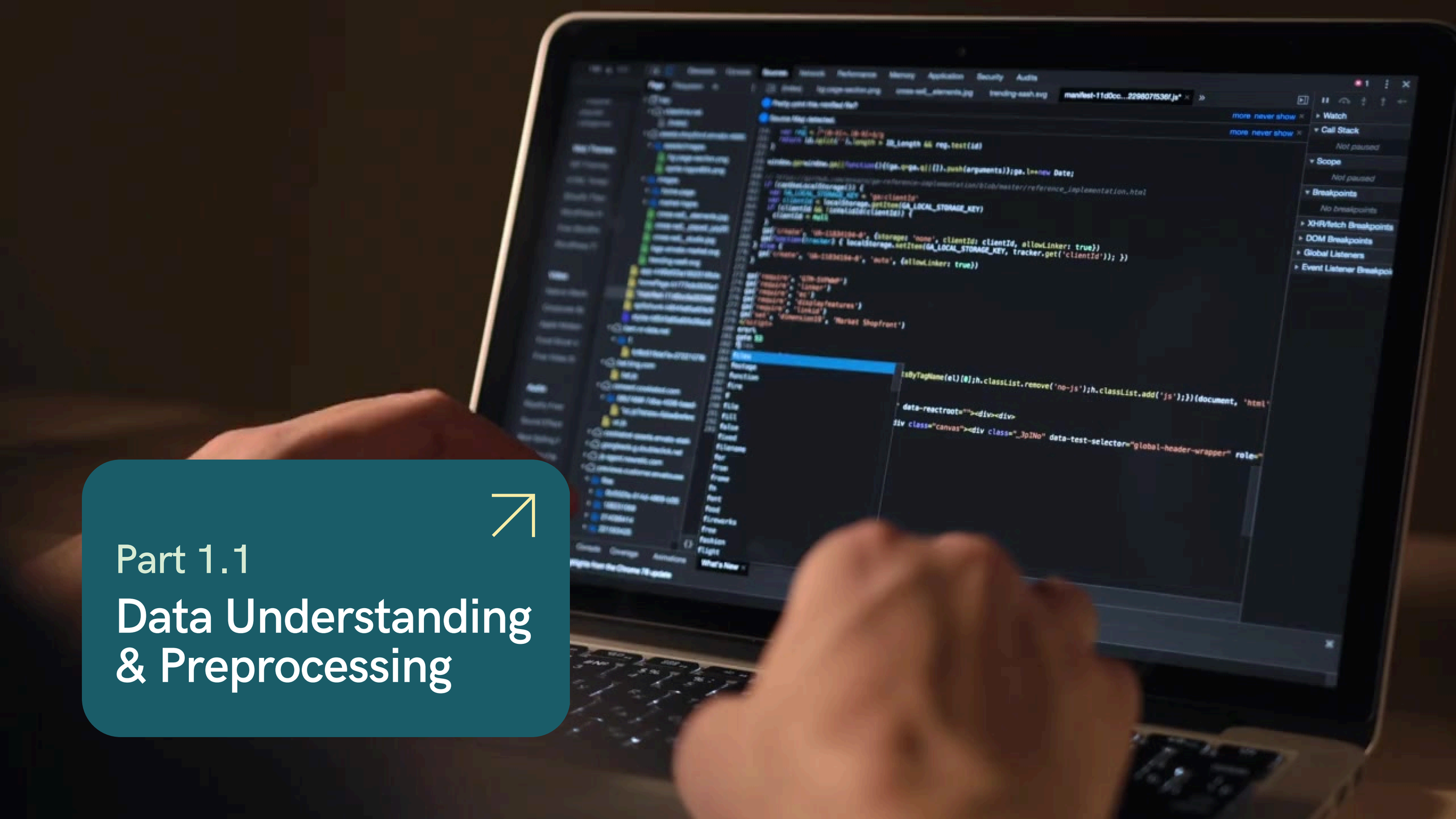


WHY MACHINE LEARNING?

1. Machine learning models can handle large and growing datasets efficiently
2. Machine learning models can be updated with new data
3. Machine learning can consider multiple variables simultaneously

WHY REGRESSION MODELS?

1. Regression models can quantify relationship between dependent variables & independent variables
2. Effective in making numerical predictions & continuous data, such as bike rental demand
3. Relatively simple to implement & understand as it come with well-established metrics for evaluation

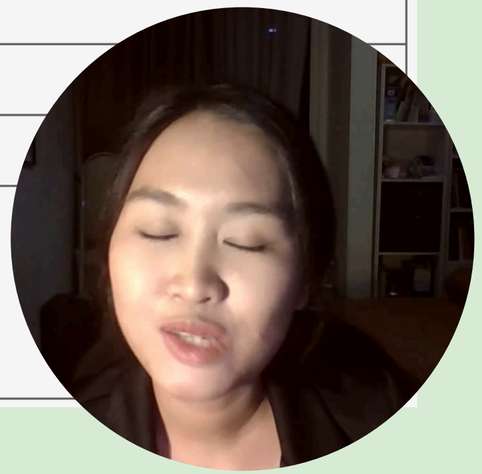


Part 1.1

Data Understanding & Preprocessing

DATASET OVERVIEW

No.	Column	Description
1	dteday	Date
2	season	Season (1: winter, 2: spring, 3: summer, 4: fall)
3	hr	Hour (0 to 23)
4	holiday	Holiday or not
5	temp	Normalized temperature in Celsius. The values are derived via (t-tmin)/(tmax-tmin), tmin=-8, tmax=+39 (hourly)
6	atemp	Normalized feeling temperature in Celsius. The values are derived via (t-tmin)/(tmax-tmin), tmin=-16, tmax=+50 (hourly)
7	hum	Normalized humidity. The values are divided into 100 (max)
8	casual	Count of casual users
9	registered	Count of registered users
10	cnt	Count of total rental bikes including both casual and registered
11	weathersit	1: Clear, Few clouds, Partly cloudy, Partly cloudy 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog



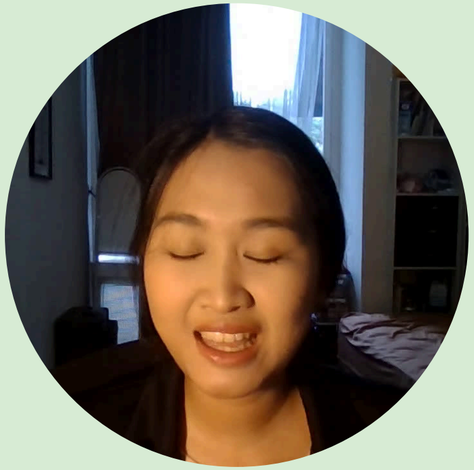
MISSING DATA

Each column **doesn't have any null values**, as indicated by the non-null count matching the number of rows (as shown by df.info()). This means there is no need to fill in any missing values.

This can be further confirmed by using the *isnull* method.

[225]
```python
if df.isnull().any().any():
    print("There are missing values in the dataset.")
else:
    print("No missing values found in the dataset.")
```
Python

```
No missing values found in the dataset.
```



DUPLICATE DATA

It's crucial to check for duplicates as they can significantly impact the outcome of the analysis.

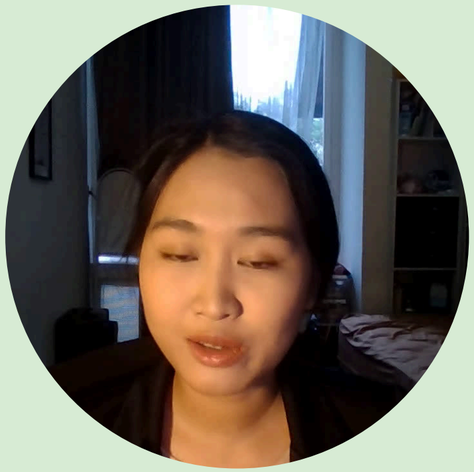
```
[224]
if df.duplicated().any():
    print("There are duplicate records in the dataset.")
else:
    print("No duplicate records found in the dataset.")
```

Python

```
...  No duplicate records found in the dataset.
```

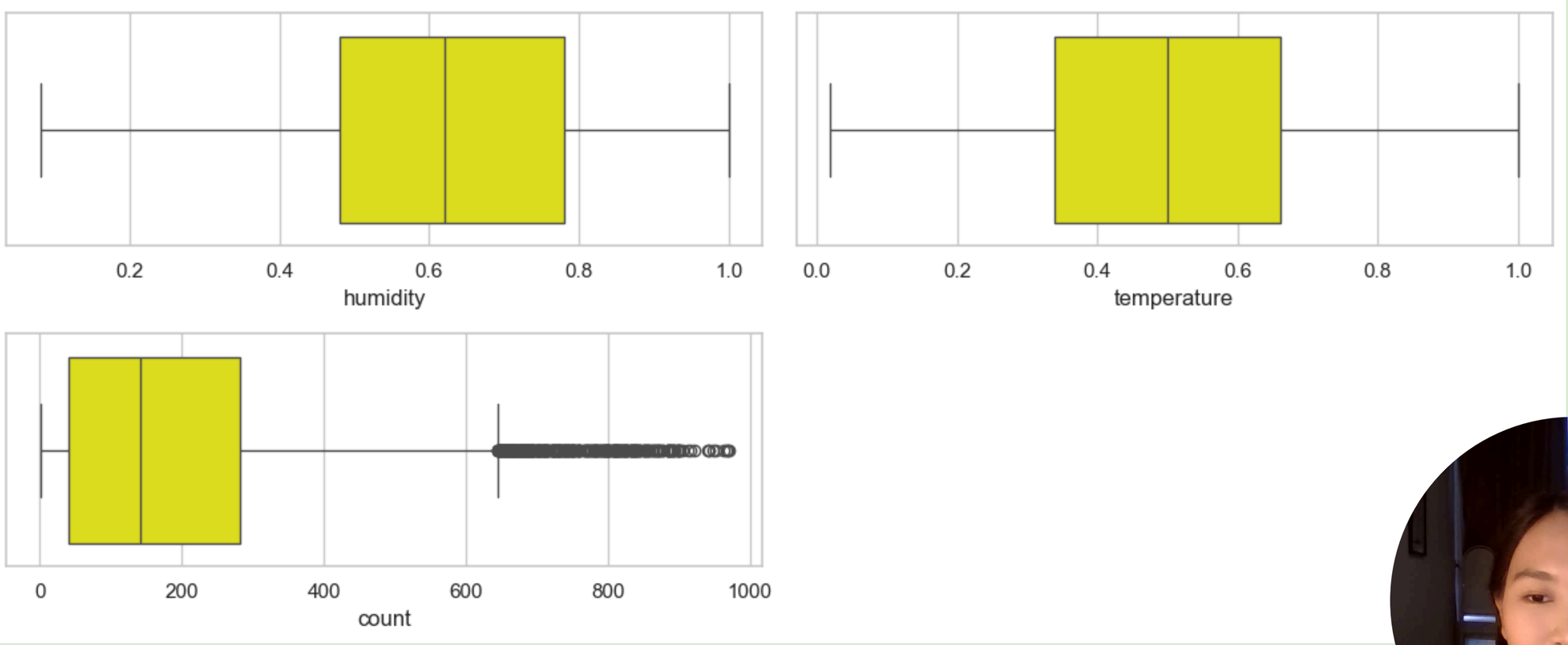
This means we don't need to do anything with the duplicate data.

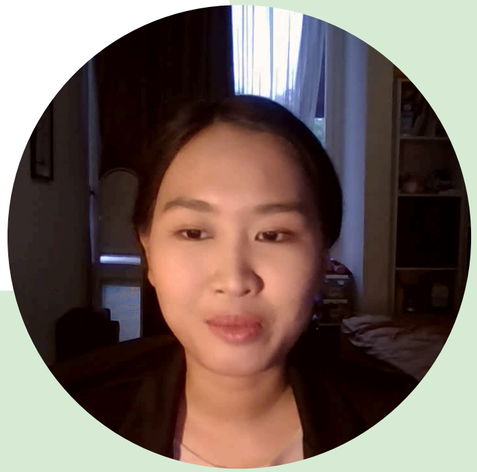
UNIQUE VALUES



	Column Name	Number of Unique Values	Sample of Unique Values
0	dteday	731	[2011-12-09, 2012-06-17, 2011-06-15, 2012-03-31, 2012-07-31, 2012-10-01, 2012-08-28, 2012-01-01, 2011-10-13, 2012-06-09, 2011-06-28, 2011-12-21, 2012-01-31, 2012-02-07, 2012-05-15, 2011-01-04, 2012-02-27, 2012-08-21, 2011-02-13, 2012-07-09, 2012-12-18, 2011-06-07, 2012-09-26, 2012-08-20, 2011-02-17, 2012-05-08, 2012-06-26, 2011-09-02, 2011-03-29, 2012-06-16, 2011-09-17, 2011-07-10, 2011-02-02, 2012-09-23, 2011-02-25, 2011-03-12, 2011-05-09, 2012-08-26, 2011-03-31, 2012-03-22, 2012-08-24, 2012-05-31, 2012-09-08, 2011-06-25, 2011-10-09, 2011-05-05, 2012-05-14, 2012-07-22, 2011-04-02, 2012-02-04, 2012-10-22, 2012-02-23, 2012-03-08, 2011-07-23, 2012-01-17, 2012-06-22, 2012-10-06, 2012-10-15, 2011-04-07, 2012-01-03, 2011-07-18, 2011-10-19, 2011-12-26, 2011-08-10, 2011-11-07, 2012-12-17, 2012-03-02, 2011-02-26, 2011-08-22, 2011-03-02, 2012-01-28, 2011-10-16, 2011-12-02, 2012-02-18, 2012-11-16, 2011-11-06, 2012-05-18, 2011-04-22, 2011-01-30, 2011-08-06, 2011-07-30, 2011-01-28, 2011-02-18,...
1	hum	89	[0.62, 0.64, 0.53, 0.87, 0.55, 0.72, 0.54, 0.93, 1.0, 0.24, 0.78, 0.6, 0.38, 0.52, 0.8, 0.37, 0.3, 0.83, 0.94, 0.61, 0.73, 0.35, 0.41, 0.74, 0.49, 0.33, 0.44, 0.77, 0.89, 0.88, 0.66, 0.7, 0.71, 0.63, 0.57, 0.43, 0.39, 0.45, 0.48, 0.5, 0.34, 0.19, 0.36, 0.65, 0.81, 0.79, 0.27, 0.29, 0.67, 0.75, 0.76, 0.31, 0.4, 0.28, 0.69, 0.08, 0.32, 0.51, 0.46, 0.59, 0.58, 0.23, 0.84, 0.82, 0.47, 0.18, 0.25, 0.42, 0.86, 0.0, 0.68, 0.56, 0.16, 0.21, 0.97, 0.85, 0.26, 0.22, 0.2, 0.17, 0.1, 0.15, 0.13, 0.9, 0.92, 0.96, 0.91, 0.12, 0.14]
2	weathersit	4	[1, 2, 3, 4]
3	holiday	2	[0, 1]
4	season	4	[4, 2, 3, 1]
5	atemp	65	[0.3485, 0.5152, 0.6212, 0.697, 0.4545, 0.6515, 0.2727, 0.6061, 0.4394, 0.2576, 0.5455, 0.2273, 0.6667, 0.4091, 0.3939, 0.6364, 0.5303, 0.5, 0.3636, 0.7424, 0.3333, 0.4242, 0.5758, 0.4697, 0.5909, 0.7576, 0.6818, 0.303, 0.7727, 0.1212, 0.803, 0.3182, 0.2121, 0.7879, 0.197, 0.1515, 0.4848, 0.1818, 0.3788, 0.0909, 0.2879, 0.2424, 0.5606, 0.1667, 0.7121, 0.7273, 0.0303, 0.8333, 0.8636, 0.8788, 0.1364, 0.1061, 0.8485, 0.8182, 0.0455, 0.8939, 0.9242, 0.0152, 0.0758, 0.0606, 0.9545, 0.9091, 0.0, 1.0, 0.9848]
6	temp	50	[0.36, 0.54, 0.62, 0.76, 0.46, 0.7, 0.26, 0.82, 0.66, 0.44, 0.58, 0.28, 0.22, 0.4, 0.38, 0.68, 0.56, 0.72, 0.52, 0.6, 0.34, 0.42, 0.64, 0.24, 0.48, 0.8, 0.32, 0.16, 0.74, 0.88, 0.3, 0.14, 0.9, 0.18, 0.06, 0.2, 0.5, 0.08, 0.78, 0.84, 0.04, 0.86, 0.12, 0.94, 0.1, 0.92, 0.96, 0.02, 0.98, 1.0]
7	hr	24	[16, 4, 23, 8, 18, 0, 22, 9, 5, 7, 14, 15, 21, 20, 11, 3, 13, 19, 6, 12, 1, 2, 10, 17]
8	casual	305	[24, 2, 17, 19, 99, 6, 20, 13, 219, 1, 11, 9, 0, 110, 51, 7, 10, 45, 30, 87, 5, 15, 60, 49, 122, 254, 81, 80, 220, 46, 36, 14, 3, 48, 31, 310, 32, 72, 26, 21, 142, 245, 16, 8, 58, 126, 47, 62, 12, 93, 27, 74, 28, 55, 50, 132, 68, 75, 29, 18, 168, 57, 61, 283, 148, 43, 4, 138, 264, 71, 121, 41, 186, 44, 39, 237, 225, 226, 224, 118, 22, 170, 162, 35, 54, 84, 200, 116, 117, 38, 53, 65, 37, 25, 196, 78, 113, 66, 33, 253, ...]
9	registered	742	[226, 16, 90, 126, 758, 39, 196, 27, 5, 315, 20, 278, 273, 127, 74, 48, 192, 110, 223, 652, 432, 808, 188, 119, 32, 157, 118, 141, 49, 134, 21, 227, 437, 50, 101, 23, 212, 1, 142, 171, 474, 8, 83, 233, 155, 370, 41, 43, 4, 88, 156, 99, 169, 400, 237, 7, 72, 59, 109, 228, 17, 148, 6, 26, 73, 100, 247, 91, 293, 95, 222, 383, 168, 79, 176, 146, 216, 369, 220, 22, 338, 3, 11, 123, 180, 112, 42, 181, 618, 209, 374, 467, 128, 179, 67, 104, 33, 413, 14, 19, ...]
10	cnt	830	[250, 18, 107, 145, 857, 45, 216, 40, 7, 534, 21, 280, 279, 138, 83, 65, 16, 302, 161, 230, 662, 477, 838, 275, 124, 47, 217, 131, 192, 55, 183, 28, 349, 75, 691, 182, 23, 292, 1, 34, 151, 178, 694, 14, 129, 269, 156, 624, 43, 5, 91, 204, 61, 130, 180, 202, 710, 256, 79, 123, 300, 19, 8, 78, 126, 268, 110, 373, 147, 102, 364, 628, 116, 137, 162, 263, 376, 282, 350, 30, 136, 273, 2, 115, 208, 698, 258, 448, 100, 261, 522, 311, 76, 117, 36, 481, 22, 26, 77, 157, ...]

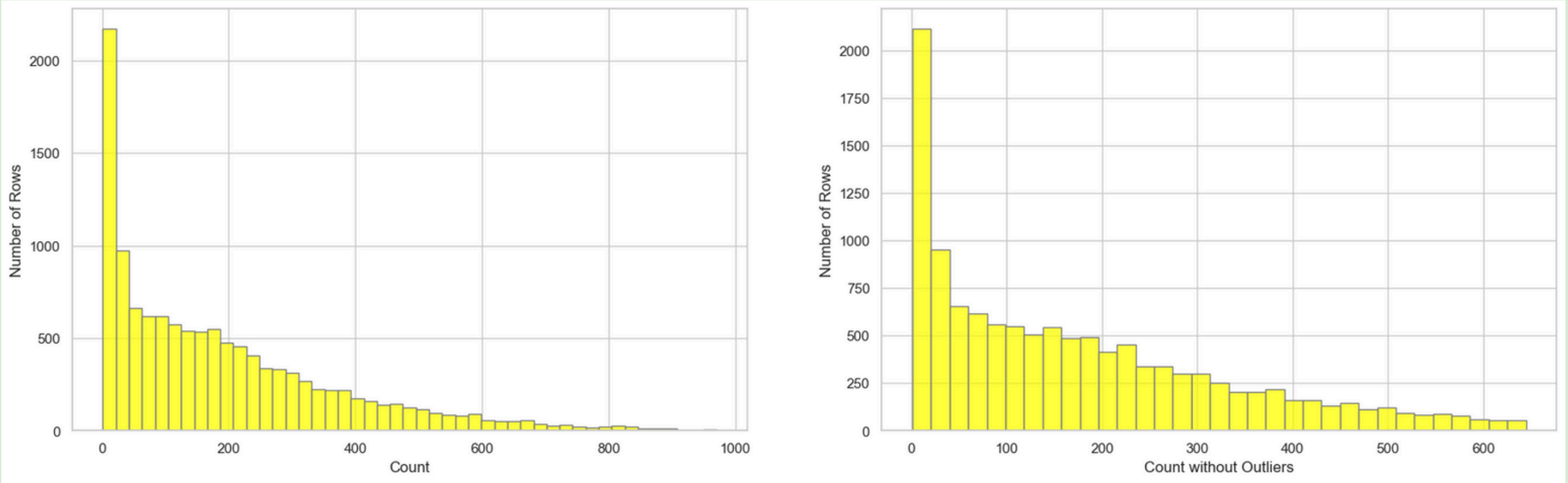
OUTLIER CHECKING

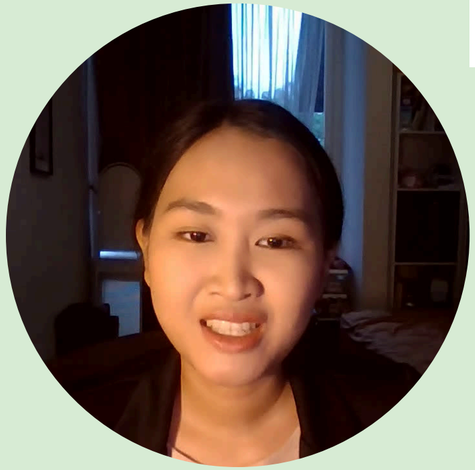




humidity have 0 outliers.
temperature have 0 outliers.
count have 338 outliers.

OUTLIER HANDLING





count has 338 outliers above the upper bound (2.78%).
count has 0 outliers below the lower bound (0.00%).
count has 338 total outliers (2.78%).

Part 1.2

Analytics

5 Models Used in This Project

1. Linear Regression
2. K-Neighbors Regressor (KNN)
3. Decision Tree Regressor
4. Random Forest Regressor
5. XGBoost Regressor

LINEAR REGRESSION

Why Use It:

- *Simplicity and Interpretability: Linear regression is straightforward to implement and interpret. It is often used as a baseline model because it provides clear insights into the relationship between the independent and dependent variables.*
- *Performance on Linear Relationships: It works well when the relationship between the features and the target variable is approximately linear.*

How It Works:

- *Linear regression finds the line of best fit by minimizing the sum of squared differences between the observed values and the predicted values.*
- *The model assumes a linear relationship between the input features (independent variables) and the output (dependent variable).*

K-NEIGHBORS REGRESSOR (KNN)

Why Use It:

- *Non-Parametric: KNN is a non-parametric method, meaning it makes no assumptions about the underlying data distribution.*
- *Flexibility: It can model complex, non-linear relationships by considering the target values of the nearest neighbors.*

How It Works:

- *KNN predicts the target value of a new data point based on the average (or weighted average) of the target values of the k-nearest neighbors in the training dataset.*
- *Distance metrics (like Euclidean distance) are used to find the nearest neighbors.*

DECISION TREE REGRESSOR

Why Use It:

- *Interpretable: Decision trees are easy to visualize and interpret, making them useful for understanding how decisions are made.*
- *Handles Non-Linear Relationships: They can capture complex, non-linear relationships between the features and the target variable.*

How It Works:

- *Decision trees split the data into subsets based on the value of the input features. The splits are chosen to minimize the variance of the target variable within each subset.*
- *The tree grows by recursively splitting subsets until a stopping criterion is met (e.g., maximum depth or minimum number of samples per leaf).*

RANDOM FOREST REGRESSOR

Why Use It:

- *Improves Overfitting: By averaging multiple decision trees, random forests reduce the risk of overfitting.*
- *Robustness: They are robust to noise and can handle missing values and outliers effectively.*

How It Works:

- *A random forest builds multiple decision trees (typically using bootstrapped samples of the training data) and averages their predictions.*
- *Each tree in the forest is built using a random subset of features, introducing more diversity and reducing overfitting.*

XGBOOST REGRESSOR

Why Use It:

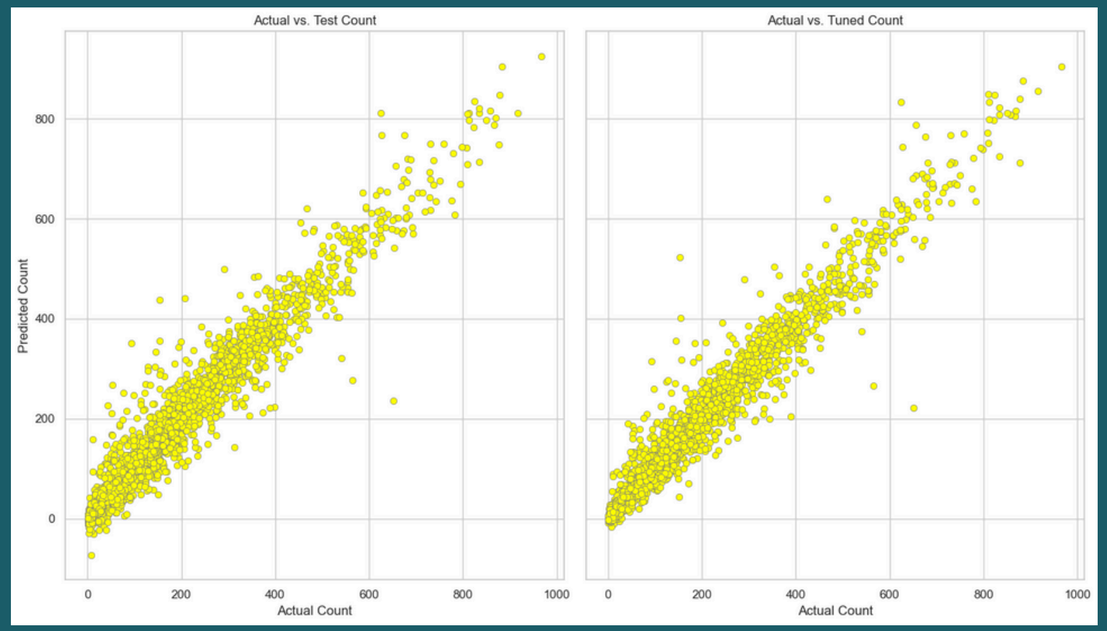
- *High Performance: XGBoost often achieves state-of-the-art results in regression tasks due to its efficiency and effectiveness.*
- *Feature Importance: It provides insights into feature importance, helping to understand which features are most influential in making predictions.*

How It Works:

- *XGBoost (Extreme Gradient Boosting) builds an ensemble of decision trees sequentially, where each tree corrects the errors of the previous one.*
- *It uses gradient descent to minimize the loss function, making it a powerful tool for optimizing complex, non-linear relationships.*

EVALUATION METRICS

Metric	Definition
Mean Absolute Error (MAE)	This metric calculates the average absolute differences between the predicted and observed values, offering a straightforward measure of prediction accuracy that is less sensitive to outliers.
Mean Absolute Percentage Error (MAPE)	This metric measures the average absolute percentage differences between predicted and observed values, providing a percentage-based measure of prediction accuracy that is easy to interpret and compare across different datasets.
R-Squared (R^2)	This metric represents the proportion of variance in the dependent variable that is predictable from the independent variables, offering an overall measure of model fit and explanatory power.



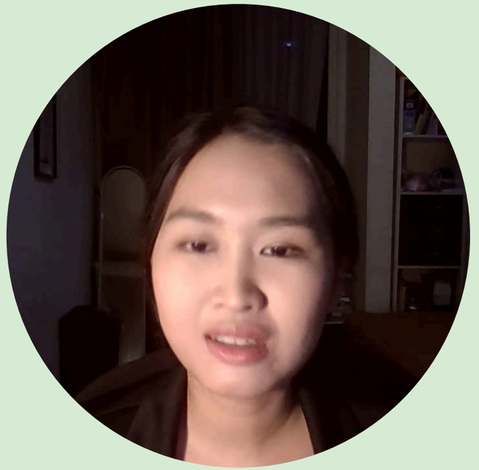
Mainly because of outliers & residuals

COMPARISON RESULT

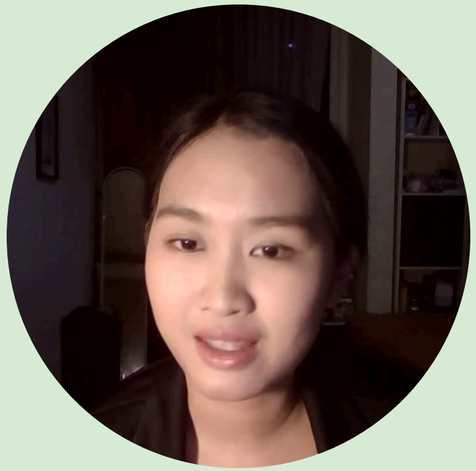
	Model	Mean_MAE	Std_MAE	Mean_MAPE	Std_MAPE	Mean_R2	Std_R2
4	XGBoost Regressor	-29.485004	0.301399	-0.468443	0.047240	0.938087	0.002991
3	RandomForest Regressor	-30.240272	0.751141	-0.359585	0.018116	0.926644	0.005165
2	DecisionTree Regressor	-38.988014	0.983649	-0.417181	0.015377	0.871336	0.003844
1	KNN Regressor	-45.319753	0.920110	-0.575683	0.029430	0.853373	0.008360
0	Linear Regression	-106.564224	0.971082	-3.412494	0.269700	0.395985	0.015018

From the results, we can conclude that:

- *XGBoost Regressor is the best performing model with the lowest mean absolute error (MAE) and highest R^2 value, indicating it has the best predictive accuracy and fits the data well (but not perfect).*
- *RandomForest Regressor also performs well as it has the lowest MAPE but has higher error and fit compared to XGBoost.*
- *DecisionTree Regressor shows moderate performance with higher errors and lower fit.*
- *KNN Regressor perform poorly with significantly higher errors and lower fit to the data.*
- *Linear Regression performs the worst in all three evaluation metrics.*



MODEL TUNING



```
RandomizedSearchCV
estimator: Pipeline
  preprocessing: ColumnTransformer
    onehot: OneHotEncoder()
    robust: RobustScaler()
    remainder: passthrough
  XGBRegressor
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=42, ...)
```

[253]

```
# Score before tuning
test_model_scores
```

	MAE	MAPE	R2
XGB	26.668122	0.420596	0.943508

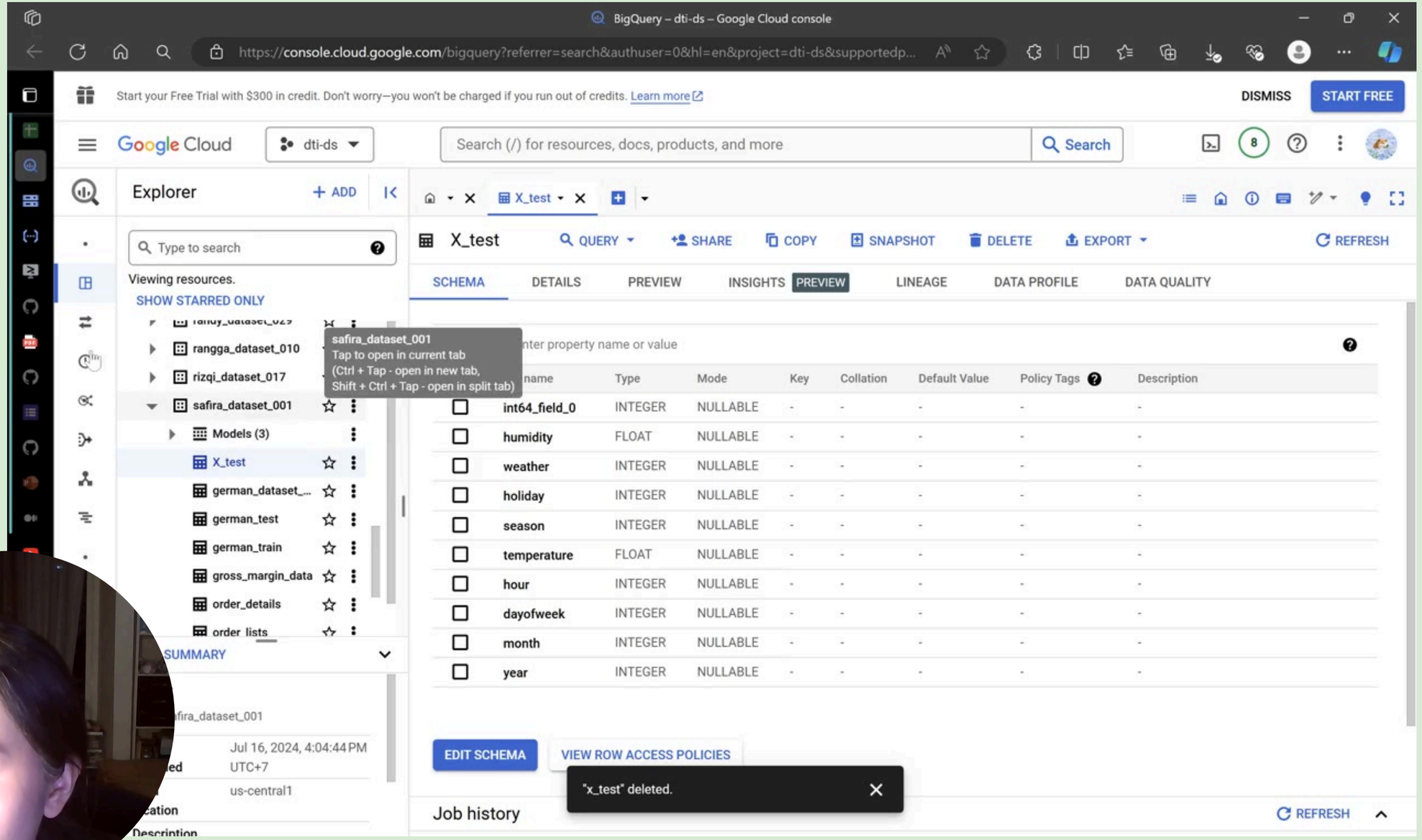
[254]

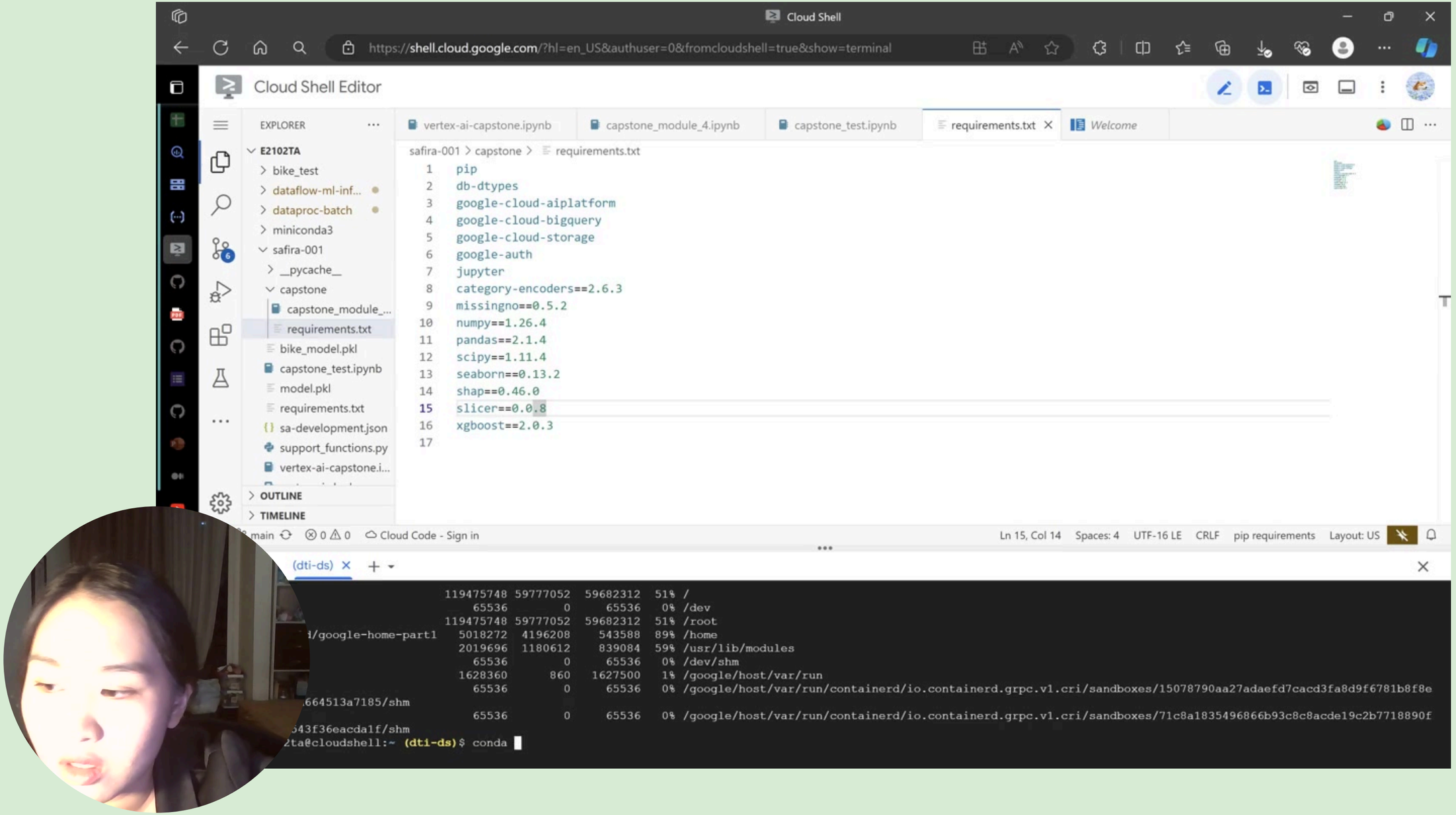
```
# Score after tuning
score_after_tuning
```

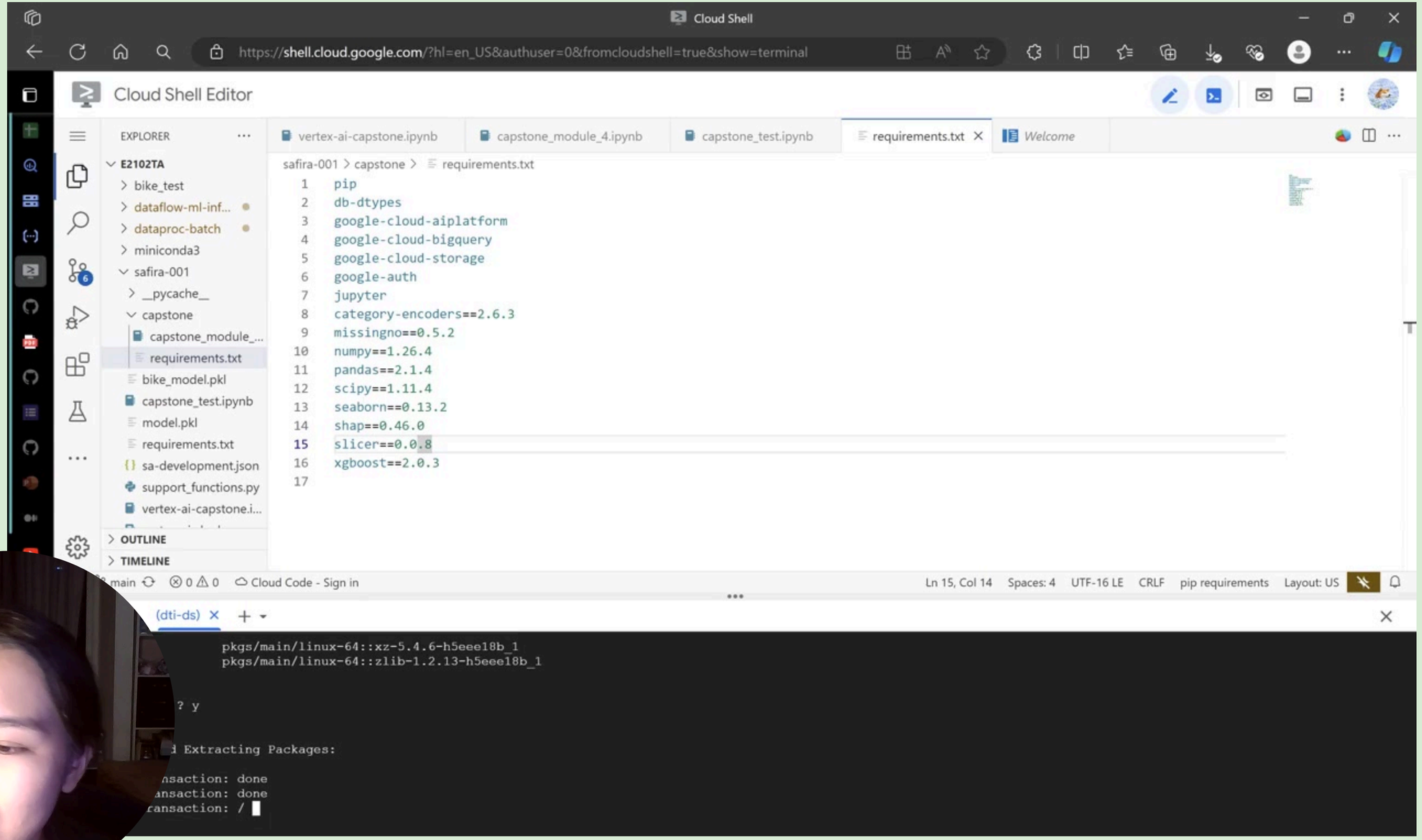
	MAE	MAPE	R2
XGB	23.068109	0.312705	0.953785

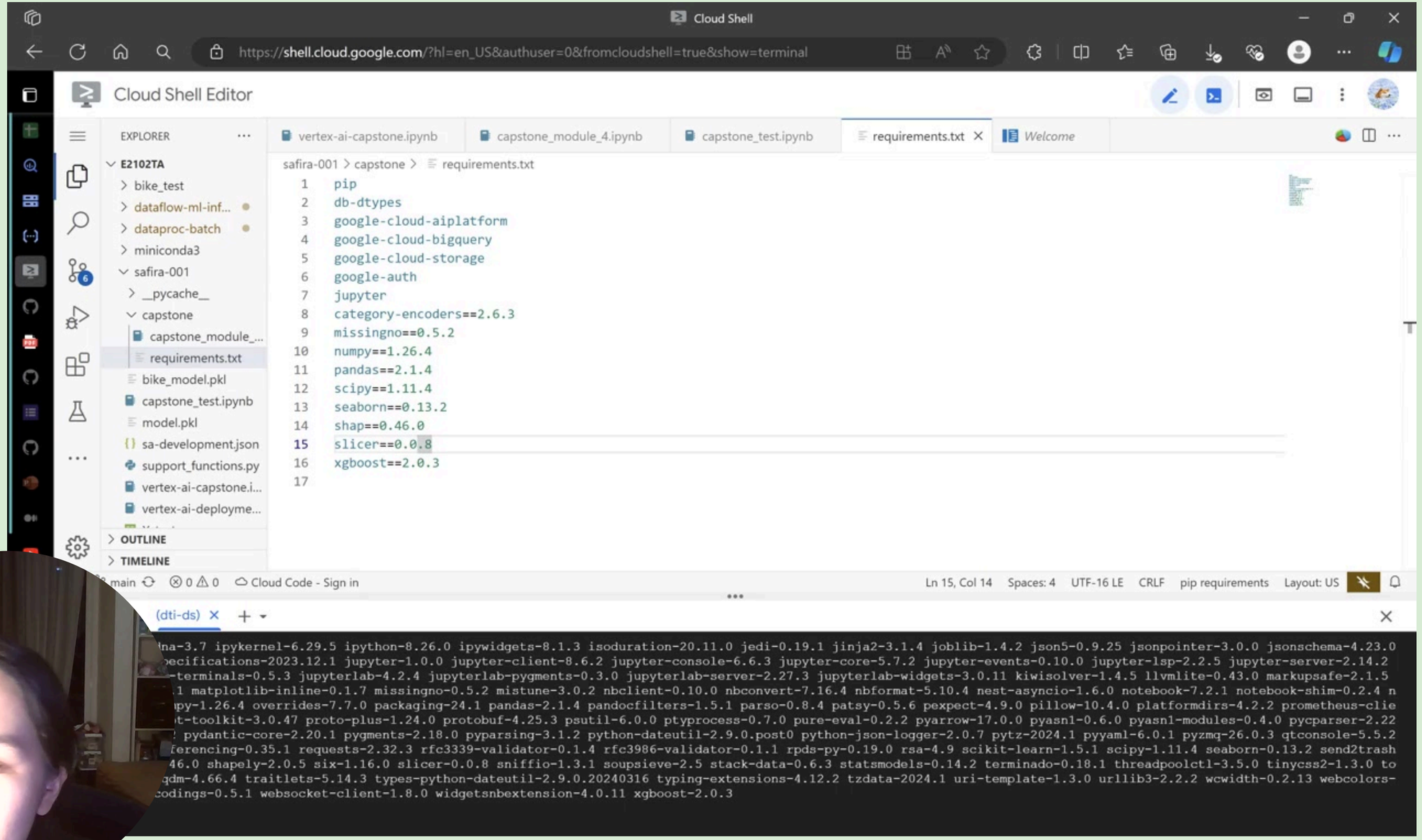
Part 2

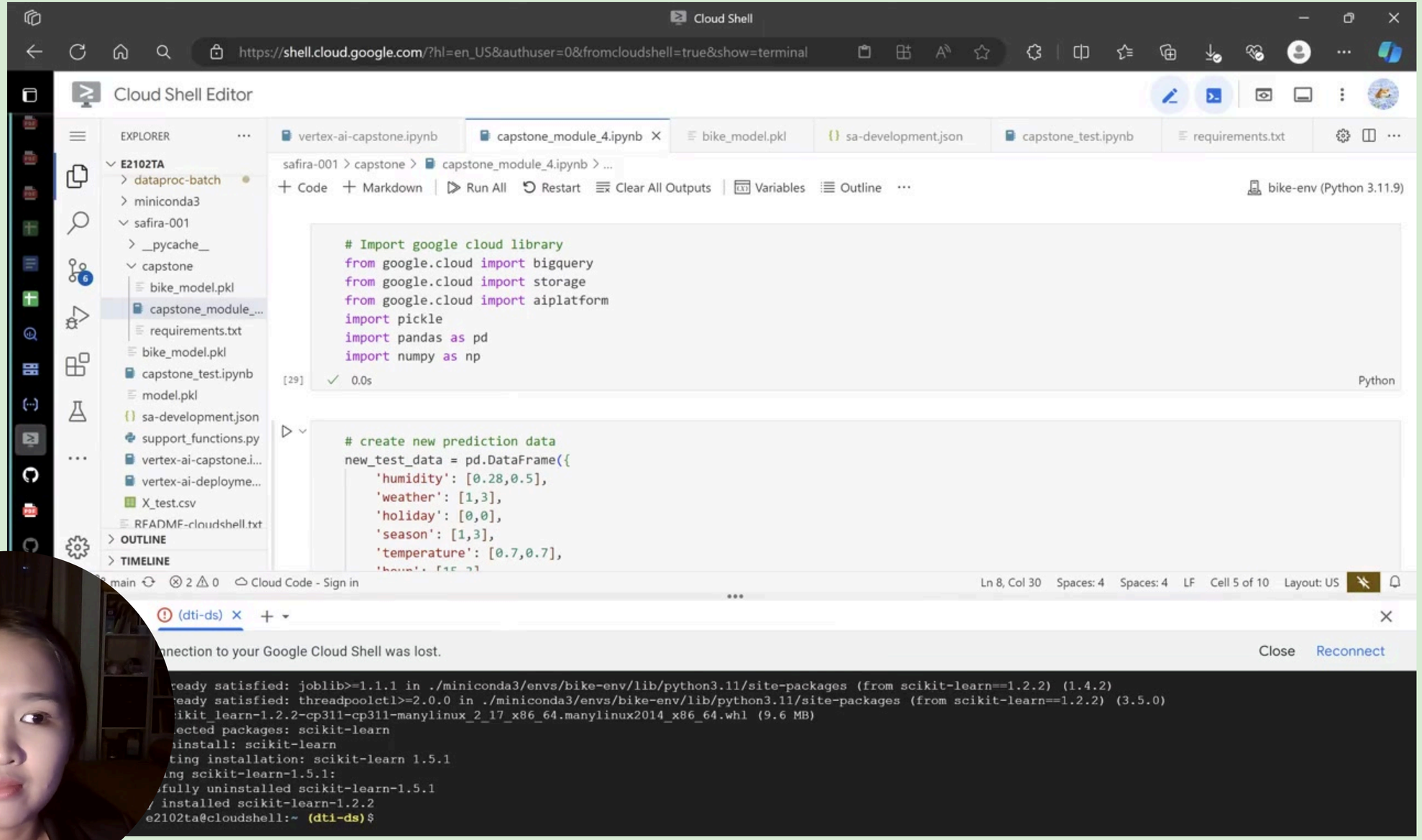
Machine Learning Model Deployment

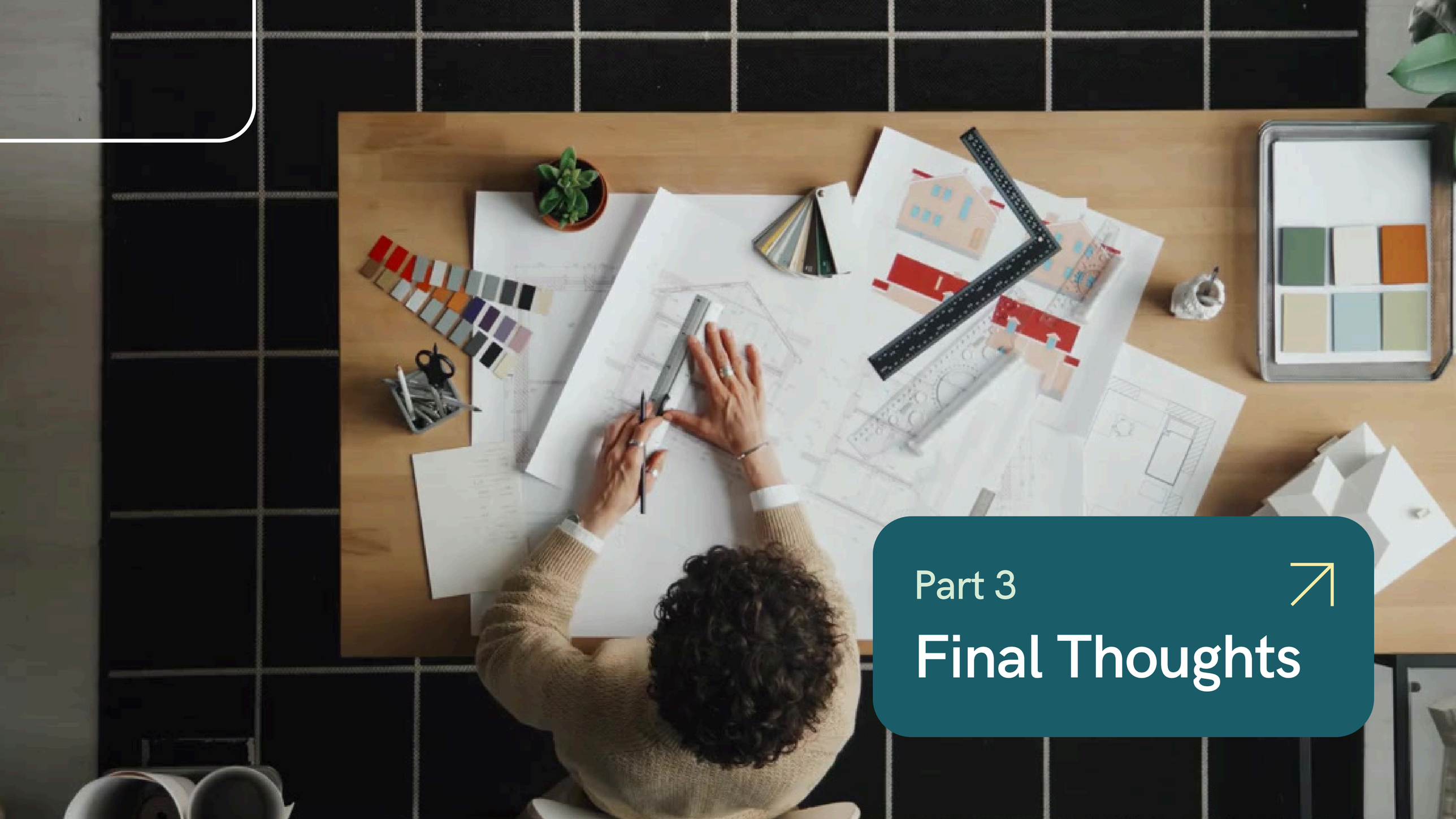

BigQuery – dti-ds – Google Cloud console
https://console.cloud.google.com/bigquery?referrer=search&authuser=0&hl=en&project=dti-ds&supportedp...
Start your Free Trial with $300 in credit. Don't worry—you won't be charged if you run out of credits. Learn more
DISMISS
START FREE
Google Cloud
dti-ds
Search (/) for resources, docs, products, and more
Search
Explorer
+ ADD
Type to search
Viewing resources.
SHOW STARRED ONLY
rangga_dataset_010
rizqi_dataset_017
safira_dataset_001
Tap to open in current tab
(Ctrl + Tap - open in new tab,
Shift + Ctrl + Tap - open in split tab)
Models (3)
X_test
german_dataset_...
german_test
german_train
gross_margin_data
order_details
order_lists
SUMMARY
X_test
QUERY
SHARE
COPY
SNAPSHOT
DELETE
EXPORT
REFRESH
SCHEMA
DETAILS
PREVIEW
INSIGHTS PREVIEW
LINEAGE
DATA PROFILE
DATA QUALITY
Enter property name or value
Name
Type
Mode
Key
Collation
Default Value
Policy Tags
Description
int64_field_0 INTEGER NULLABLE
humidity FLOAT NULLABLE
weather INTEGER NULLABLE
holiday INTEGER NULLABLE
season INTEGER NULLABLE
temperature FLOAT NULLABLE
hour INTEGER NULLABLE
dayofweek INTEGER NULLABLE
month INTEGER NULLABLE
year INTEGER NULLABLE
EDIT SCHEMA
VIEW ROW ACCESS POLICIES
"x_test" deleted.
Job history
REFRESH
Jul 16, 2024, 4:04:44 PM UTC+7
us-central1


Cloud Shell
https://shell.cloud.google.com/?hl=en_US&authuser=0&fromcloudshell=true&show=terminal
Cloud Shell Editor
EXPLORER
E2102TA
bike_test
dataflow-ml-inf...
dataproc-batch
miniconda3
safira-001
__pycache__
capstone
capstone_module_...
requirements.txt
bike_model.pkl
capstone_test.ipynb
model.pkl
requirements.txt
sa-development.json
support_functions.py
vertex-ai-capstone.i...
OUTLINE
TIMELINE
vertex-ai-capstone.ipynb
capstone_module_4.ipynb
capstone_test.ipynb
requirements.txt
Welcome
safira-001 > capstone > requirements.txt
1 pip
2 db-dtypes
3 google-cloud-aiplatform
4 google-cloud-bigquery
5 google-cloud-storage
6 google-auth
7 jupyter
8 category-encoders==2.6.3
9 missingno==0.5.2
10 numpy==1.26.4
11 pandas==2.1.4
12 scipy==1.11.4
13 seaborn==0.13.2
14 shap==0.46.0
15 slicer==0.0.8
16 xgboost==2.0.3
17
main
0 0
Cloud Code - Sign in
Ln 15, Col 14
Spaces: 4
UTF-16 LE
CRLF
pip requirements
Layout: US
(dti-ds)
119475748 59777052 59682312 51% /
65536 0 65536 0% /dev
119475748 59777052 59682312 51% /root
5018272 4196208 543588 89% /home
2019696 1180612 839084 59% /usr/lib/modules
65536 0 65536 0% /dev/shm
1628360 860 1627500 1% /google/host/var/run
65536 0 65536 0% /google/host/var/run/containerd/io.containerd.grpc.v1.cri/sandboxes/15078790aa27adaefd7cacd3fa8d9f6781b8f8e
65536 0 65536 0% /google/host/var/run/containerd/io.containerd.grpc.v1.cri/sandboxes/71c8a1835496866b93c8c8acde19c2b7718890f
(dti-ds)$ conda


Cloud Shell
https://shell.cloud.google.com/?hl=en_US&authuser=0&fromcloudshell=true&show=terminal
Cloud Shell Editor
EXPLORER
E2102TA
bike_test
dataflow-ml-inf...
dataproc-batch
miniconda3
safira-001
__pycache__
capstone
capstone_module_...
requirements.txt
bike_model.pkl
capstone_test.ipynb
model.pkl
requirements.txt
sa-development.json
support_functions.py
vertex-ai-capstone.i...
OUTLINE
TIMELINE
vertex-ai-capstone.ipynb
capstone_module_4.ipynb
capstone_test.ipynb
requirements.txt
Welcome
safira-001 > capstone > requirements.txt
1 pip
2 db-dtypes
3 google-cloud-aiplatform
4 google-cloud-bigquery
5 google-cloud-storage
6 google-auth
7 jupyter
8 category-encoders==2.6.3
9 missingno==0.5.2
10 numpy==1.26.4
11 pandas==2.1.4
12 scipy==1.11.4
13 seaborn==0.13.2
14 shap==0.46.0
15 slicer==0.0.8
16 xgboost==2.0.3
17
main
0 0
Cloud Code - Sign in
Ln 15, Col 14
Spaces: 4
UTF-16 LE
CRLF
pip requirements
Layout: US
(dti-ds)
pkgs/main/linux-64::xz-5.4.6-h5eee18b_1
pkgs/main/linux-64::zlib-1.2.13-h5eee18b_1
? y
Extracting Packages:
nsaction: done
ansaction: done
ransaction: /


Cloud Shell
https://shell.cloud.google.com/?hl=en_US&authuser=0&fromcloudshell=true&show=terminal
Cloud Shell Editor
EXPLORER
E2102TA
bike_test
dataflow-ml-inf...
dataproc-batch
miniconda3
safira-001
__pycache__
capstone
capstone_module_...
requirements.txt
bike_model.pkl
capstone_test.ipynb
model.pkl
sa-development.json
support_functions.py
vertex-ai-capstone.i...
vertex-ai-deployme...
OUTLINE
TIMELINE
vertex-ai-capstone.ipynb
capstone_module_4.ipynb
capstone_test.ipynb
requirements.txt
Welcome
safira-001 > capstone > requirements.txt
1 pip
2 db-dtypes
3 google-cloud-aiplatform
4 google-cloud-bigquery
5 google-cloud-storage
6 google-auth
7 jupyter
8 category-encoders==2.6.3
9 missingno==0.5.2
10 numpy==1.26.4
11 pandas==2.1.4
12 scipy==1.11.4
13 seaborn==0.13.2
14 shap==0.46.0
15 slicer==0.0.8
16 xgboost==2.0.3
17
main
0 0
Cloud Code - Sign in
Ln 15, Col 14
Spaces: 4
UTF-16 LE
CRLF
pip requirements
Layout: US
(dti-ds)
ipykernel-6.29.5 ipython-8.26.0 ipywidgets-8.1.3 isoduration-20.11.0 jedi-0.19.1 jinja2-3.1.4 joblib-1.4.2 json5-0.9.25 jsonpointer-3.0.0 jsonschema-4.23.0
jupyter-1.0.0 jupyter-client-8.6.2 jupyter-console-6.6.3 jupyter-core-5.7.2 jupyter-events-0.10.0 jupyter-lsp-2.2.5 jupyter-server-2.14.2
jupyterlab-4.2.4 jupyterlab-pygments-0.3.0 jupyterlab-server-2.27.3 jupyterlab-widgets-3.0.11 kiwisolver-1.4.5 llvmlite-0.43.0 markupsafe-2.1.5
matplotlib-inline-0.1.7 missingno-0.5.2 mistune-3.0.2 nbclient-0.10.0 nbconvert-7.16.4 nbformat-5.10.4 nest-asyncio-1.6.0 notebook-7.2.1 notebook-shim-0.2.4
overrides-7.7.0 packaging-24.1 pandas-2.1.4 pandocfilters-1.5.1 parso-0.8.4 patsy-0.5.6 pexpect-4.9.0 pillow-10.4.0 platformdirs-4.2.2 prometheus-clie
proto-plus-1.24.0 protobuf-4.25.3 psutil-6.0.0 ptyprocess-0.7.0 pure-eval-0.2.2 pyarrow-17.0.0 pyasn1-0.6.0 pyasn1-modules-0.4.0 pycparser-2.22
pydantic-core-2.20.1 pygments-2.18.0 pyparsing-3.1.2 python-dateutil-2.9.0.post0 python-json-logger-2.0.7 pytz-2024.1 pyyaml-6.0.1 pyzmq-26.0.3 qtconsole-5.5.2
requests-2.32.3 rfc3339-validator-0.1.4 rfc3986-validator-0.1.1 rpds-py-0.19.0 rsa-4.9 scikit-learn-1.5.1 scipy-1.11.4 seaborn-0.13.2 send2trash
shapely-2.0.5 six-1.16.0 slicer-0.0.8 sniffio-1.3.1 soupsieve-2.5 stack-data-0.6.3 statsmodels-0.14.2 terminado-0.18.1 threadpoolctl-3.5.0 tinycss2-1.3.0
traitlets-5.14.3 types-python-dateutil-2.9.0.20240316 typing-extensions-4.12.2 tzdata-2024.1 uri-template-1.3.0 urllib3-2.2.2 wcwidth-0.2.13 webcolors-
websocket-client-1.8.0 widgetsnbextension-4.0.11 xgboost-2.0.3


Cloud Shell
https://shell.cloud.google.com/?hl=en_US&authuser=0&fromcloudshell=true&show=terminal
Cloud Shell Editor
EXPLORER
E2102TA
dataproc-batch
miniconda3
safira-001
__pycache__
capstone
bike_model.pkl
capstone_module_...
requirements.txt
bike_model.pkl
capstone_test.ipynb
model.pkl
sa-development.json
support_functions.py
vertex-ai-capstone.i...
vertex-ai-deployme...
X_test.csv
README-cloudshell.txt
OUTLINE
TIMELINE
vertex-ai-capstone.ipynb
capstone_module_4.ipynb
bike_model.pkl
sa-development.json
capstone_test.ipynb
requirements.txt
safira-001 > capstone > capstone_module_4.ipynb > ...
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
bike-env (Python 3.11.9)
# Import google cloud library
from google.cloud import bigquery
from google.cloud import storage
from google.cloud import aiplatform
import pickle
import pandas as pd
import numpy as np
[29]
0.0s
Python
# create new prediction data
new_test_data = pd.DataFrame({
'humidity': [0.28,0.5],
'weather': [1,3],
'holiday': [0,0],
'season': [1,3],
'temperature': [0.7,0.7],
main
2 0
Cloud Code - Sign in
Ln 8, Col 30
Spaces: 4
Spaces: 4
LF
Cell 5 of 10
Layout: US
(dti-ds)
nection to your Google Cloud Shell was lost.
Close
Reconnect
eady satisfied: joblib>=1.1.1 in ./miniconda3/envs/bike-env/lib/python3.11/site-packages (from scikit-learn==1.2.2) (1.4.2)
eady satisfied: threadpoolctl>=2.0.0 in ./miniconda3/envs/bike-env/lib/python3.11/site-packages (from scikit-learn==1.2.2) (3.5.0)
ikit_learn-1.2.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (9.6 MB)
ected packages: scikit-learn
install: scikit-learn
ting installation: scikit-learn 1.5.1
ng scikit-learn-1.5.1:
fully uninstalled scikit-learn-1.5.1
installed scikit-learn-1.2.2
e2102ta@cloudshell:~ (dti-ds)$



Part 3

Final Thoughts

TRANSLATING DEMAND INTO REVENUE

Rental Price Per Hour = $2

Cost per Bike per Month = $50

Fixed Cost per Month = $1000

Variable Cost per Month = $10

Current Bike Supply = *100*

Total Revenue from Predicted Demand: $882218.00

Operational Costs: $96000.00

Median Predicted Bike Demand: 141.0

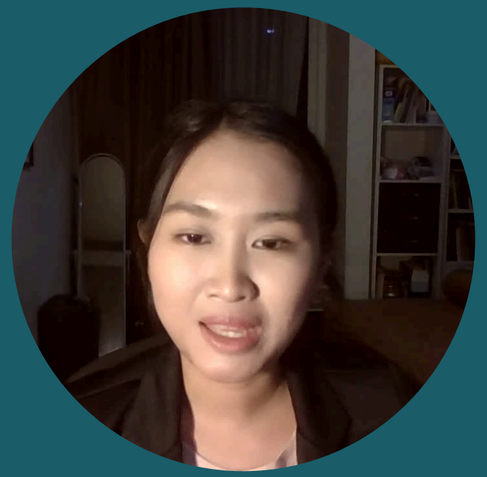
Cost of Adding New Bikes: $98400.00

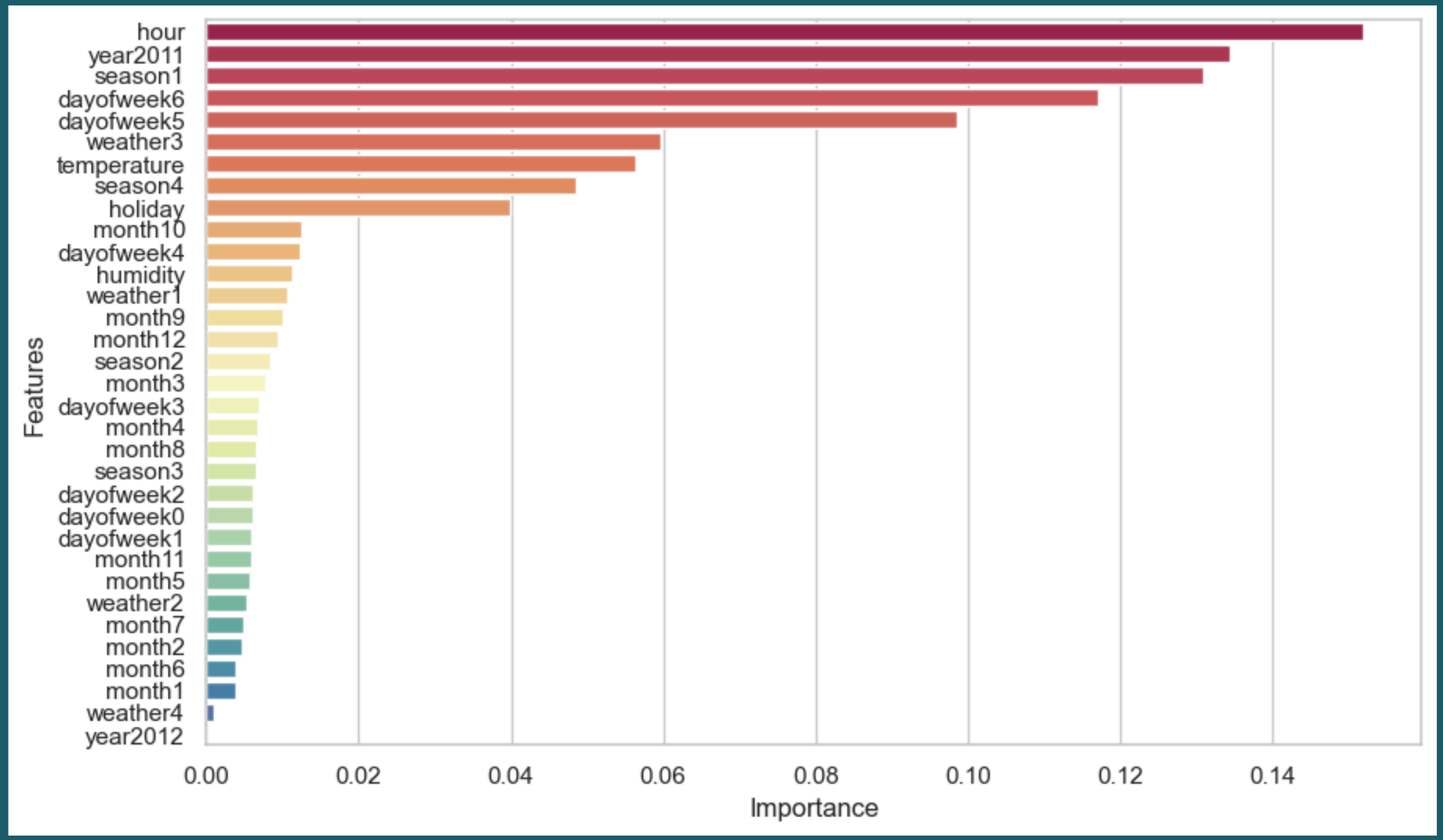
Total Costs: $194400.00

Profit: $687818.00

Break-Even Point: 97200.00 hourly rentals within 2 years or 4050.00 daily rentals

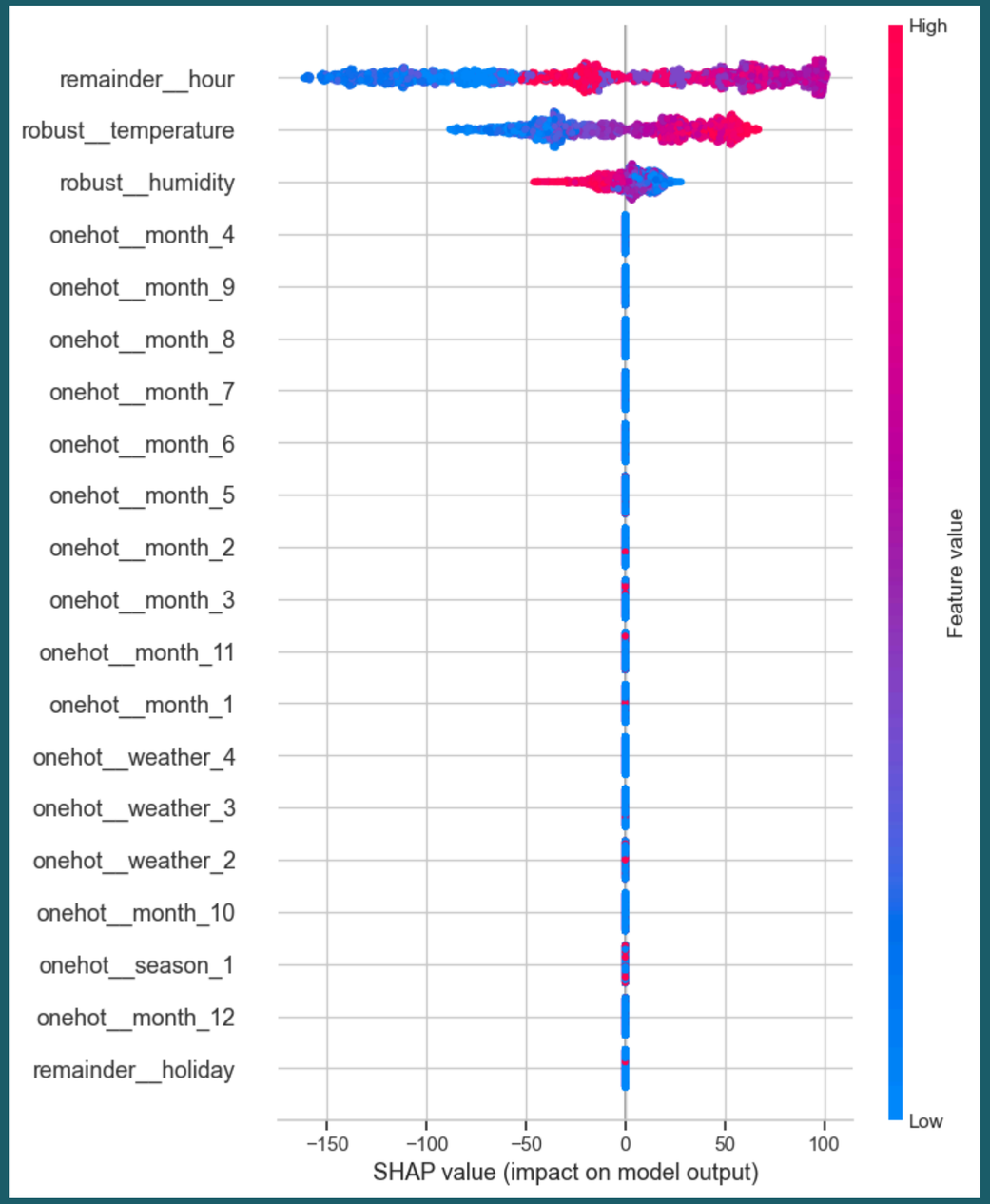
FACTORS THAT INFLUENCE BIKE RENTAL DEMAND





FACTORS THAT INFLUENCE BIKE RENTAL DEMAND

SHAP (SHapley Additive exPlanations) is a game theoretic approach to explain the output of any machine learning model.



CONCLUSION

1. Model Performance: *The predictive model, using a tuned XGBoost algorithm, has demonstrated strong performance with evaluation metrics: Mean Absolute Error (MAE) of 23.07, Mean Absolute Percentage Error (MAPE) of 0.31, and an R^2 value of 0.95. These metrics suggest that the model provides accurate predictions, though it is not without limitations.*

2. Demand Prediction: *The model is effective in forecasting the number of bikes needed, but the results should be interpreted with caution due to the imbalance in the dataset. This imbalance may impact the accuracy of predictions and should be considered when making decisions based on the model's output.*

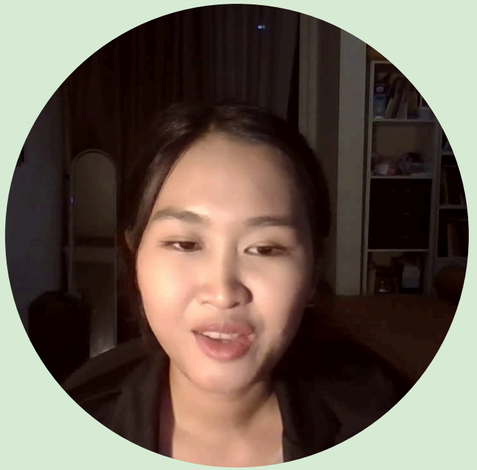
3. Profitability: *Adding new bikes as predicted by the model could potentially increase profitability, provided that the actual rental rates exceed the break-even point. This indicates a positive financial impact, contingent on achieving higher-than-break-even rentals.*

4. Key Features: *Hour, year 2011, season 1, day of the week 6, and weather 3 are identified as the most influential features affecting the model's predictions. These factors play a significant role in determining bike demand and should be considered in operational decisions.*

RECOMMENDATION

1. Update Data: *To enhance the accuracy of predictions, it is recommended to incorporate more recent data, preferably from 2024. This will help capture current demand trends and improve the model's relevance to the present context.*

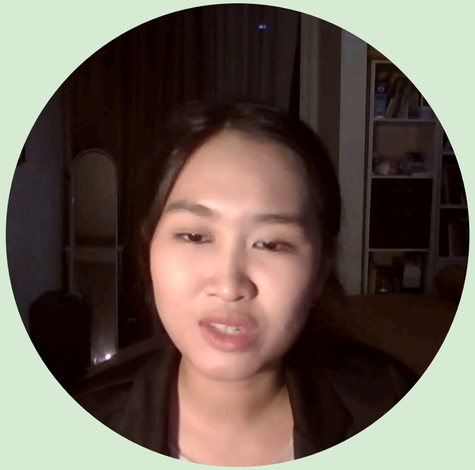
2. Feature Expansion: *Adding additional features that capture more recent trends or variables not previously considered could provide a more comprehensive view of bike demand. This may lead to more accurate and actionable insights.*



3. Monitor and Validate: *Continuously monitor real-world rental data and compare it against model predictions. This will help validate the model's effectiveness and adjust strategies as needed to ensure that the predicted increase in profitability aligns with actual outcomes.*

4. Survey: *To know the effect before and after the new bike addition.*

LIMITATIONS



1. Data Imbalance: *The dataset used for training the model is imbalanced, which may affect the accuracy and generalizability of the predictions. This limitation should be addressed by obtaining more balanced data or using techniques to mitigate the impact of imbalance.*

2. Historical Data: *The use of historical data from 2011-2012 may not fully reflect current market conditions or demand patterns. This historical data might not account for recent changes in user behavior or external factors affecting bike rentals.*

3. Model Constraints: *While the XGBoost model performs well, it may not capture all nuances of bike demand. Further model refinement and validation are necessary to ensure robustness and reliability in varying conditions.*

https://github.com/fabilia/bike-sharing/tree/main