Transit Demand Forecasting

Firoz Kabir

Agenda

- Problem Statement
- Data Collection
- EDA
- Naive Model
- Additional Data
- Enhanced Model
- Model Evaluation
- Conclusions

Problem Statement

- Forecast Transit Demand using XGBoost
- Use Toronto Ferry Ticket Redemption Data
- Add Weather Data and Holiday Indicator
- Model (XGBoost) typically not used on this kind of data
- Novel application of Data Centric ML + XGBoost

Agenda

- ~~Problem Statement~~
- **Data Collection**
- EDA
- Naive Model
- Additional Data
- Enhanced Model
- Model Evaluation
- Conclusions

Data Collection

DATA PREVIEW

_id	Timestamp	Redemption Count	Sales Count
1	2025-03-18T15:45:00	20	0
2	2025-03-18T15:30:00	49	54
3	2025-03-18T15:15:00	13	31

DATA FEATURES

Column	Description
_id	Unique row identifier for Open Data database
Timestamp	The 15 minute interval (end time) that the data represents
Redemption Count	The ticket redemption count between the previous and current timestamp
Sales Count	The ticket sales count between the previous and current timestamp

DATA QUALITY

Last refreshed
2025-03-18

Overall score
100%

Grade
Gold

Daily Data Report for March 2025

TORONTO INTL A
ONTARIO
Current Station Operator: NAVCAN

Latitude:	43°40'36.000" N	**Longitude**:	79°37'50.000" W	**Elevation**:	173.40 m
Climate ID:	6158731	**WMO ID**:	71624	**TC ID**:	YYZ

Related Data

No related data is available for this station

Additional Search Options

Nearby Stations with Data

Historical Data Search

Download Data

Daily Data (2025)

CSV XML Metadata(txt)

Download Data

Get More Data

Sources:

- https://open.toronto.ca/dataset/toronto-island-ferry-ticket-counts/
- https://climate.weather.gc.ca/climate_data/daily_data_e.html
- Holidays python package (https://pypi.org/project/holidays/)

Data Collection

Download files to local if running or the first time

```
[4]: download_files = False
```

```
[5]: def download_weather_data(start_year=2016,
                          end_year=2024,
                          station_id=51459):

    local_path = f".{os.sep}data{os.sep}weather-data{os.sep}"
    print(f"Weather data will be saved in {local_path}")
    Path(local_path).mkdir(parents=True, exist_ok=True)

    for year in range(start_year, end_year+1):

        for month in range(12):
            url = f"https://climate.weather.gc.ca/climate_data/bulk_data_e.html?format=csv&stationID=51459&Year={year}&Month={month+1}&Day=1&timeframe=1&submit= Download+Data"
            file = f"{local_path}{os.sep}{year}_{month+1}.csv"
            print(f"Downloading weather data for year: {year}, month: {month+1} into file: {file}")
            response = r.get(url)
            df = pd.read_csv(io.StringIO(response.content.decode('utf-8')))
            df.to_csv(file, index=False, header=True, quoting=csv.QUOTE_ALL)

    print("== done ==")
```

```
[6]: def download_ferry_ticket_data():
    url = "https://ckan0.cf.opendata.inter.prod-toronto.ca/dataset/toronto-island-ferry-ticket-counts/resource/c46719f5-8006-44e1-8b1e-5ad90bb9f6f4/download/Toronto%20Island%20Ferry%20
    file = f".{os.sep}data{os.sep}toronto_island_ferry_ticket_counts.csv"
    print(f"Downloading Toronto ferry traffic data from Toronto Open Data portal into file: {file}")
    response = r.get(url)
    df = pd.read_csv(io.StringIO(response.content.decode('utf-8')))
    df.to_csv(file, index=False, header=True, quoting=csv.QUOTE_ALL)
    print("== done ==")
```

```
[7]: if download_files:
    download_weather_data()
    download_ferry_ticket_data()
```

Agenda

- ~~Problem Statement~~
- ~~Data Collection~~
- **EDA**
- Naive Model
- Additional Data
- Enhanced Model
- Model Evaluation
- Conclusions

EDA - head(), info(), describe()

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 240821 entries, 2025-03-21 08:45:00 to 2015-05-01 13:30:00
Data columns (total 1 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   redemption_count  240821 non-null  int64
dtypes: int64(1)
memory usage: 3.7 MB
```

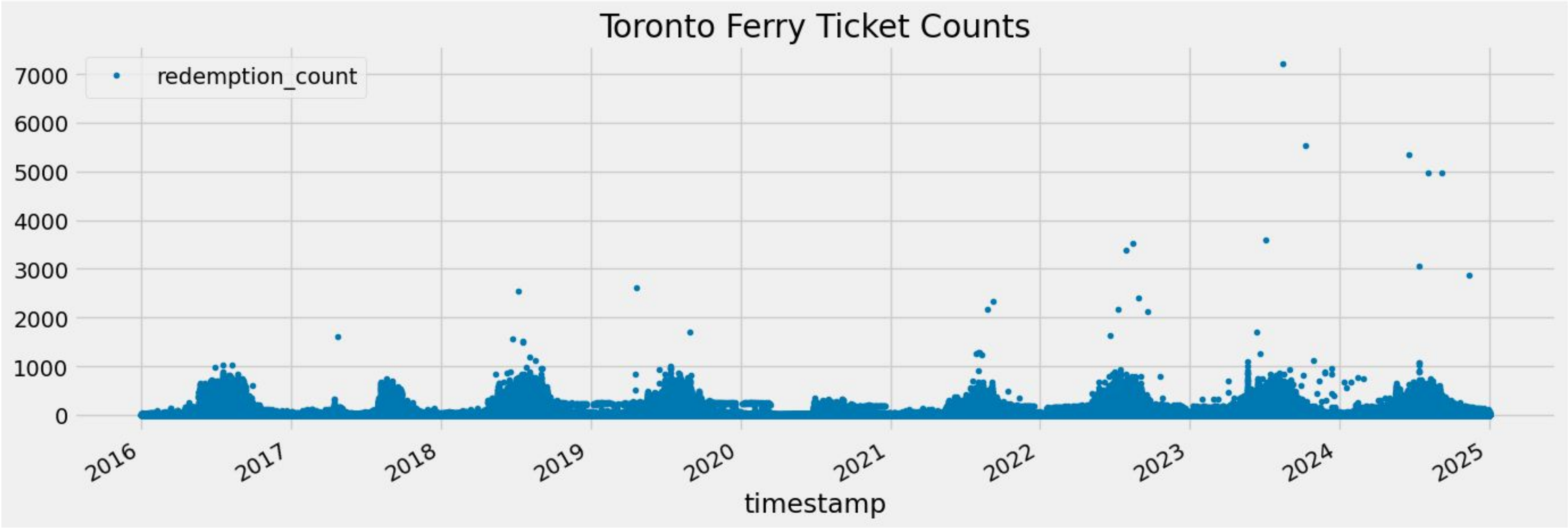
```
df.head()
```

timestamp	redemption_count
2025-03-21 08:45:00	2
2025-03-21 08:30:00	9
2025-03-21 08:15:00	32
2025-03-21 08:00:00	6
2025-03-21 07:45:00	0

```
df.describe()
```

	redemption_count
count	220937.000000
mean	46.562187
std	101.554887
min	0.000000
25%	3.000000
50%	10.000000
75%	37.000000
max	7216.000000

EDA - data distribution (partial years removed)



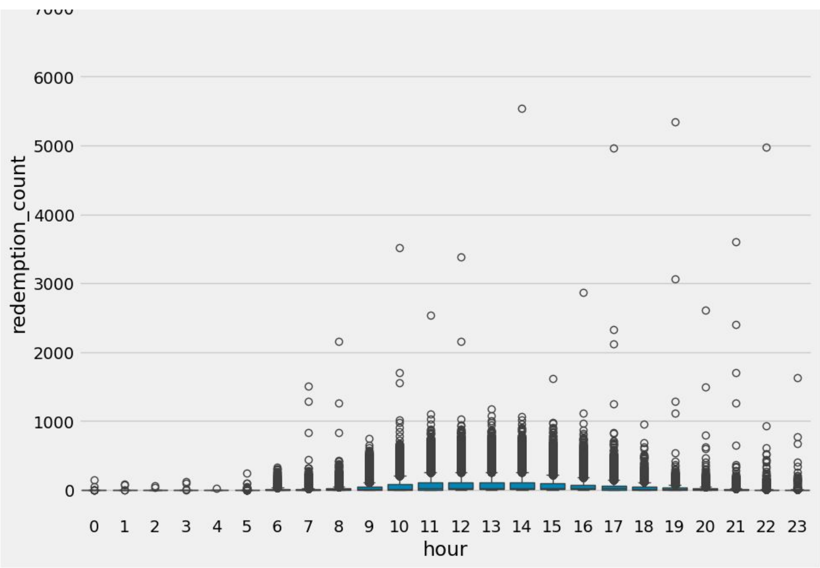
EDA - data distribution - outlier detection

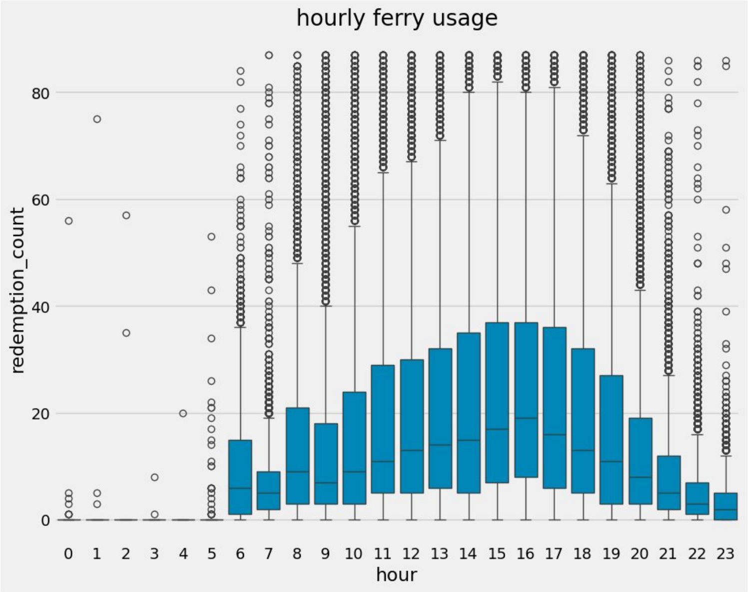
```
1]: # As we can see from the redemption_count column, there are some extreme outliers.
    # Let's try to clean that up using IQR.

    Q1 = df['redemption_count'].quantile(0.25)
    Q3 = df['redemption_count'].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5*IQR
    upper = Q3 + 1.5*IQR
```

```
2]: print(f"lower: {lower}, upper: {upper}" )

    lower: -48.0, upper: 88.0
```

```
3]: # removing outlier values outside partial years 2015 and 2025

    df = df.loc[df.redemption_count > lower]
    df = df.loc[df.redemption_count < upper]
```

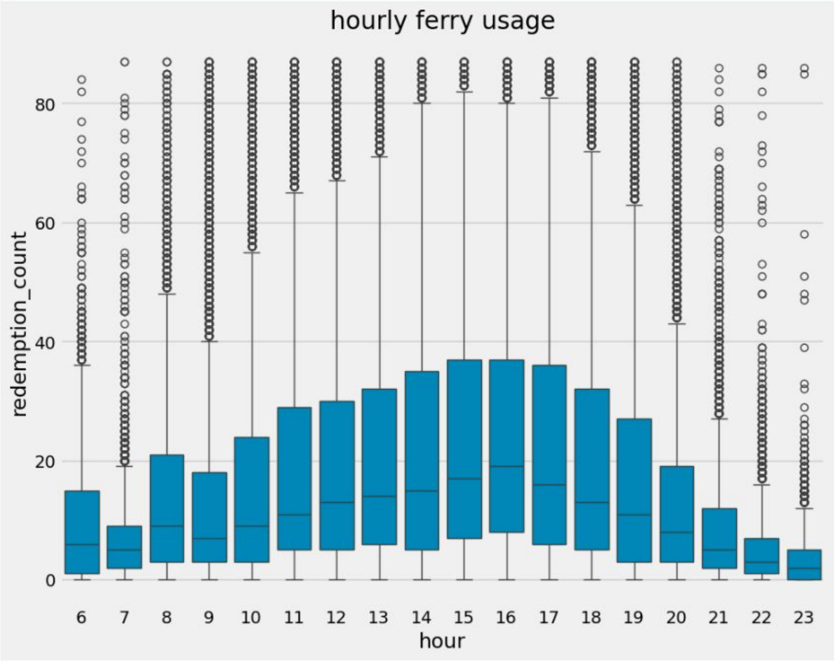
EDA - data distribution - outlier detection





EDA - data distribution - outlier detection

```python
# let's remove any records before 5am and replot
df = df.loc[df.hour > 5]
# replot after remove 0 - 5 hours
fig, ax = plt.subplots(figsize=(10, 8))
sns.boxplot(data=df, x='hour', y='redemption_count')
ax.set_title('hourly ferry usage')
plt.show()
```



Agenda

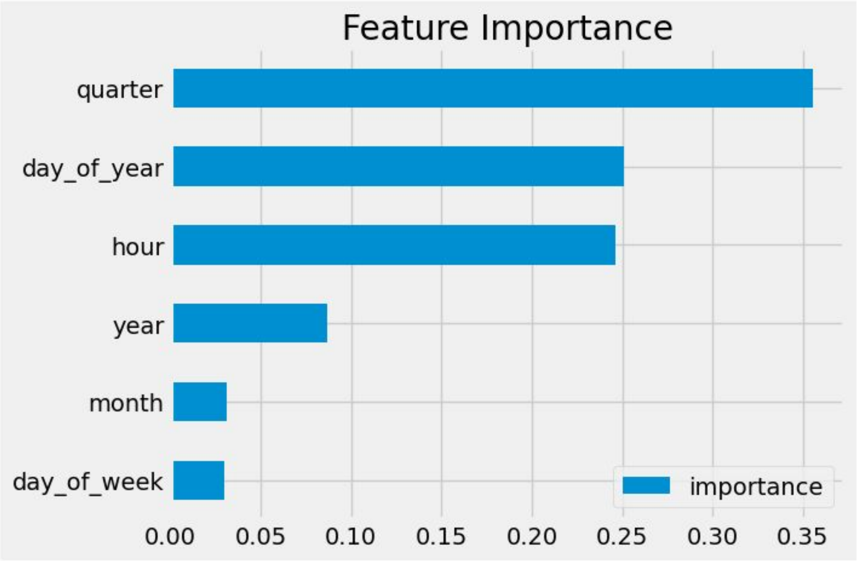
- ~~Problem Statement~~
- ~~Data Collection~~
- ~~EDA~~
- **Naive Model**
- Additional Data
- Enhanced Model
- Model Evaluation
- Conclusions

Naive Model

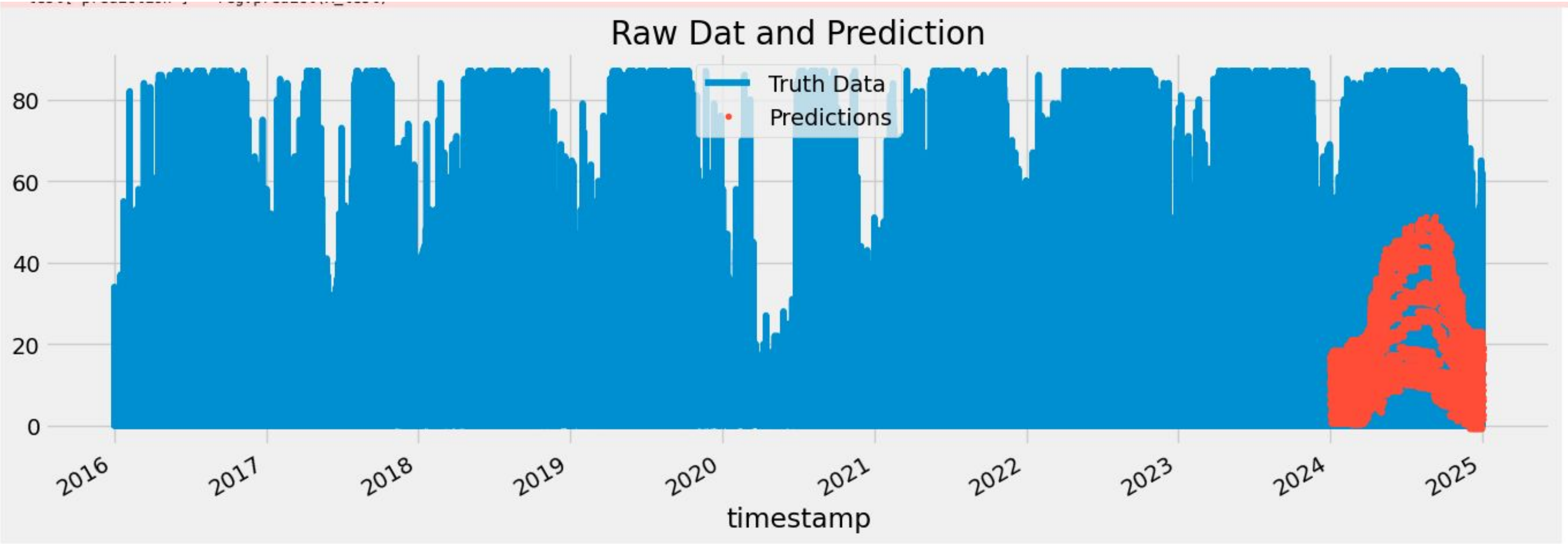
```
# training set is [2016 - 2023]
# test set is > 2023

train = df.loc[df.index < '01-01-2024']
test = df.loc[df.index >= '01-01-2024']

# fig, ax = plt.subplots(figsize=(15, 5))
# train.plot(ax=ax, label='Training Set', title='Data Train/Test Split')
# test.plot(ax=ax, label='Test Set')
# ax.axvline('01-01-2024', color='black', ls='--')
# ax.legend(['Training Set', 'Test Set'])
# plt.show()
```

```
FEATURES = ['day_of_year', 'hour', 'day_of_week', 'quarter', 'month', 'year']
TARGET = 'redemption_count'

X_train = train[FEATURES]
y_train = train[TARGET]

X_test = test[FEATURES]
y_test = test[TARGET]
```

```
reg = xgb.XGBRegressor(base_score=0.5, booster='gbtree',
                       n_estimators=1000,
                       early_stopping_rounds=50,
                       objective='reg:linear',
                       max_depth=3,
                       learning_rate=0.01)
reg.fit(X_train, y_train,
        eval_set=[(X_train, y_train), (X_test, y_test)],
        verbose=100)
```



Naive Model


Raw Dat and Prediction
Truth Data
Predictions
0
20
40
60
80
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
timestamp

Naive Model

RMSE

```
score = np.sqrt(mean_squared_error(test['redemption_count'], test['prediction']))
print(f'RMSE Score on Test set: {score:0.2f}')
```

```
RMSE Score on Test set: 14.83
```

```
test['error'] = np.abs(test[TARGET] - test['prediction'])
test['date'] = test.index.date
test.groupby(['date'])['error'].mean().sort_values(ascending=False).head(10)
```

```
date
2024-07-10    21.015122
2024-05-02    20.923778
2024-03-30    19.817264
2024-07-16    19.420949
2024-03-13    19.082908
2024-04-06    18.731630
2024-04-14    18.690246
2024-05-07    18.620986
2024-06-06    18.381669
2024-06-15    18.174068
Name: error, dtype: float64
```

MAPE

```
mean_absolute_percentage_error(test['redemption_count'], test['prediction'])
```

```
1892655722135552.0
```

Agenda

- ~~Problem Statement~~
- ~~Data Collection~~
- ~~EDA~~
- ~~Naive Mode~~
- **Additional Data**
- Enhanced Model
- Model Evaluation
- Conclusions

Additional Data

```
[42]: df_weather.columns
```

```
[42]: Index(['Longitude (x)', 'Latitude (y)', 'Station Name', 'Climate ID',
             'Date/Time (LST)', 'Year', 'Month', 'Day', 'Time (LST)', 'Flag',
             'Temp (°C)', 'Temp Flag', 'Dew Point Temp (°C)', 'Dew Point Temp Flag',
             'Rel Hum (%)', 'Rel Hum Flag', 'Precip. Amount (mm)',
             'Precip. Amount Flag', 'Wind Dir (10s deg)', 'Wind Dir Flag',
             'Wind Spd (km/h)', 'Wind Spd Flag', 'Visibility (km)',
             'Visibility Flag', 'Stn Press (kPa)', 'Stn Press Flag', 'Hmdx',
             'Hmdx Flag', 'Wind Chill', 'Wind Chill Flag', 'Weather'],
            dtype='object')
```

```
[43]: df_weather = df_weather[['Year', 'Month', 'Day', 'Time (LST)', 'Temp (°C)']]
```

```
[44]: df_weather = df_weather.rename(columns={'Year': 'year',
                                              'Month': 'month',
                                              'Day': 'day_of_month',
                                              'Time (LST)': 'hour_minute',
                                              'Temp (°C)': 'temp_c'})
```

Agenda

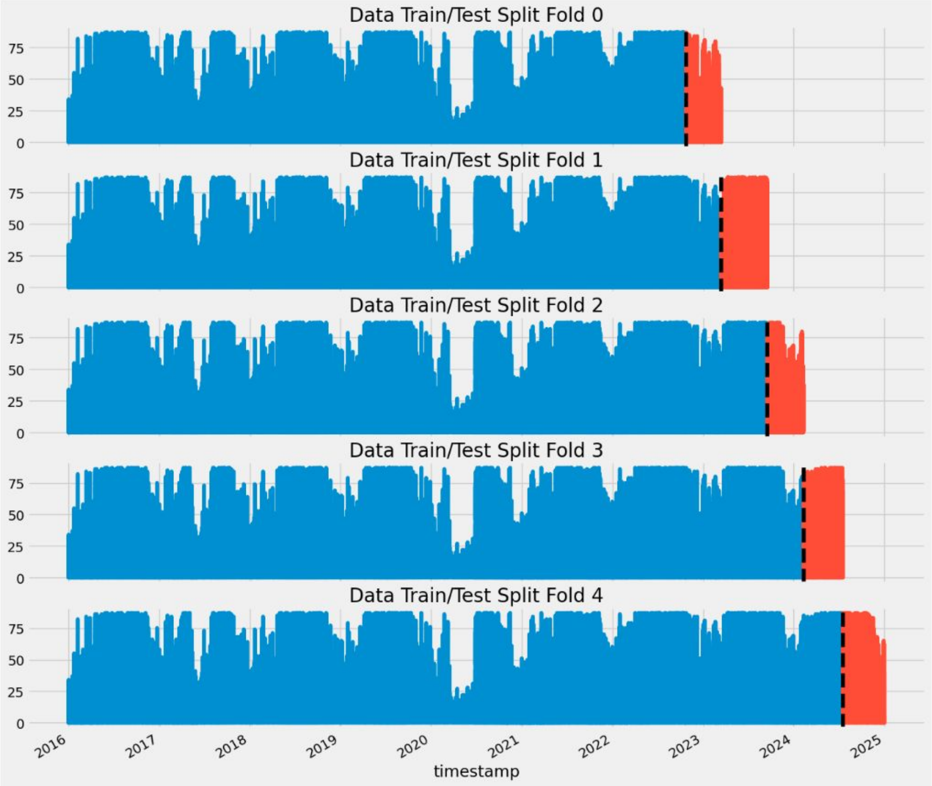
- ~~Problem Statement~~
- ~~Data Collection~~
- ~~EDA~~
- ~~Naive Mode~~
- ~~Additional Data~~
- **Enhanced Model**
- Model Evaluation
- Conclusions

TimeSeris Split

Train and Test

```
[1]: # timeseries split

tss = TimeSeriesSplit(n_splits=5, test_size=24*1*365, gap=24)
df = df.sort_index()
```

```
[2]: fig, axs = plt.subplots(5, 1, figsize=(15, 15), sharex=True)

fold = 0
for train_idx, val_idx in tss.split(df):
    train = df.iloc[train_idx]
    test = df.iloc[val_idx]
    train['redemption_count'].plot(ax=axs[fold],
                                   label='Training Set',
                                   title=f'Data Train/Test Split Fold {fold}')
    test['redemption_count'].plot(ax=axs[fold],
                                  label='Test Set')
    axs[fold].axvline(test.index.min(), color='black', ls='--')
    fold += 1
plt.show()
```



Lag Features

```
def add_lags(df, target_column):
    target_map = df[target_column].to_dict()
    df['lag_1'] = (df.index - pd.Timedelta('364 days')).map(target_map)
    df['lag_2'] = (df.index - pd.Timedelta('728 days')).map(target_map)
    df['lag_3'] = (df.index - pd.Timedelta('1092 days')).map(target_map)
    return df
```

```
#### lag features

df = add_lags(df, target_column='redemption_count')
```

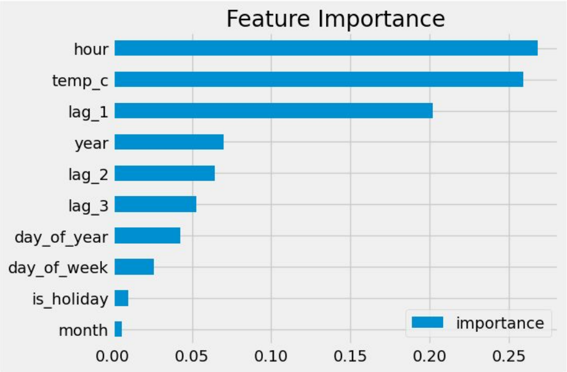
Enhanced Model

```python
FEATURES = ['day_of_year', 'hour', 'day_of_week',
             'month', 'year', 'temp_c',
            'lag_1', 'lag_2', 'lag_3',
           'is_holiday']
TARGET = 'redemption_count'
```

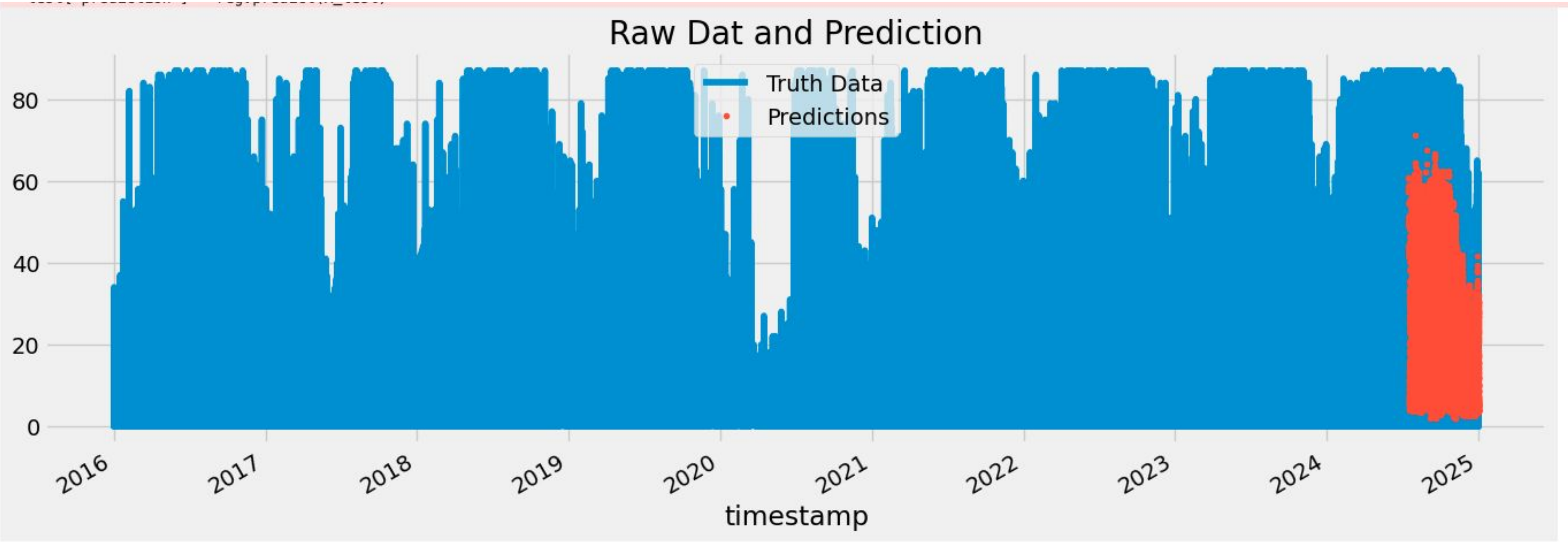
```python
print(f'Score across folds {np.mean(scores):0.4f}')
print(f'Fold scores:{scores}')
```

```
Score across folds 13.6283
Fold scores:[np.float64(11.07392973884338), np.float64(16.86877959716281), np.float64(12.079517121421649), np.float64(15.113334760179244), np.float64(13.005980656487003)]
```

```python
fi = pd.DataFrame(data=reg.feature_importances_,
             index=reg.feature_names_in_,
             columns=['importance'])
fi.sort_values('importance').plot(kind='barh', title='Feature Importance')
plt.show()
```



Enhanced Model



Enhanced Model

RMSE

```
score = np.sqrt(mean_squared_error(test['redemption_count'], test['prediction']))
print(f'RMSE Score on Test set: {score:0.2f}')
```

```
RMSE Score on Test set: 13.01
```

```
test['error'] = np.abs(test[TARGET] - test['prediction'])
test['date'] = test.index.date
test.groupby(['date'])['error'].mean().sort_values(ascending=False).head(10)
```

```
date
2024-09-06    21.410403
2024-09-24    18.908494
2024-09-25    16.795186
2024-09-09    16.126796
2024-10-14    15.936092
2024-10-31    14.991946
2024-10-01    14.721109
2024-08-17    14.621920
2024-12-29    14.600295
2024-08-18    14.468682
Name: error, dtype: float64
```

MAPE

```
mean_absolute_percentage_error(test['redemption_count'], test['prediction'])
```

```
1580030958239744.0
```

Agenda

- ~~Problem Statement~~
- ~~Data Collection~~
- ~~EDA~~
- ~~Naive Mode~~
- ~~Additional Data~~
- ~~TimeSeris Split, Lag Features and Cross Validation~~
- ~~Enhanced Model~~
- **Model Evaluation**
- Conclusions

Model Evaluation

RMSE

```
score = np.sqrt(mean_squared_error(test['redemption_count'], test['prediction']))
print(f'RMSE Score on Test set: {score:0.2f}')
```

```
RMSE Score on Test set: 14.83
```

```
test['error'] = np.abs(test[TARGET] - test['prediction'])
test['date'] = test.index.date
test.groupby(['date'])['error'].mean().sort_values(ascending=False).head(10)
```

```
date
2024-07-10    21.015122
2024-05-02    20.923778
2024-03-30    19.817264
2024-07-16    19.420949
2024-03-13    19.082908
2024-04-06    18.731630
2024-04-14    18.690246
2024-05-07    18.620986
2024-06-06    18.381669
2024-06-15    18.174068
Name: error, dtype: float64
```

MAPE

```
mean_absolute_percentage_error(test['redemption_count'], test['prediction'])
```

```
1892655722135552.0
```

RMSE

```
score = np.sqrt(mean_squared_error(test['redemption_count'], test['prediction']))
print(f'RMSE Score on Test set: {score:0.2f}')
```

```
RMSE Score on Test set: 13.01
```

```
test['error'] = np.abs(test[TARGET] - test['prediction'])
test['date'] = test.index.date
test.groupby(['date'])['error'].mean().sort_values(ascending=False).head(10)
```

```
date
2024-09-06    21.410403
2024-09-24    18.908494
2024-09-25    16.795186
2024-09-09    16.126796
2024-10-14    15.936092
2024-10-31    14.991946
2024-10-01    14.721109
2024-08-17    14.621920
2024-12-29    14.600295
2024-08-18    14.468682
Name: error, dtype: float64
```

MAPE

```
mean_absolute_percentage_error(test['redemption_count'], test['prediction'])
```

```
1580030958239744.0
```

Agenda

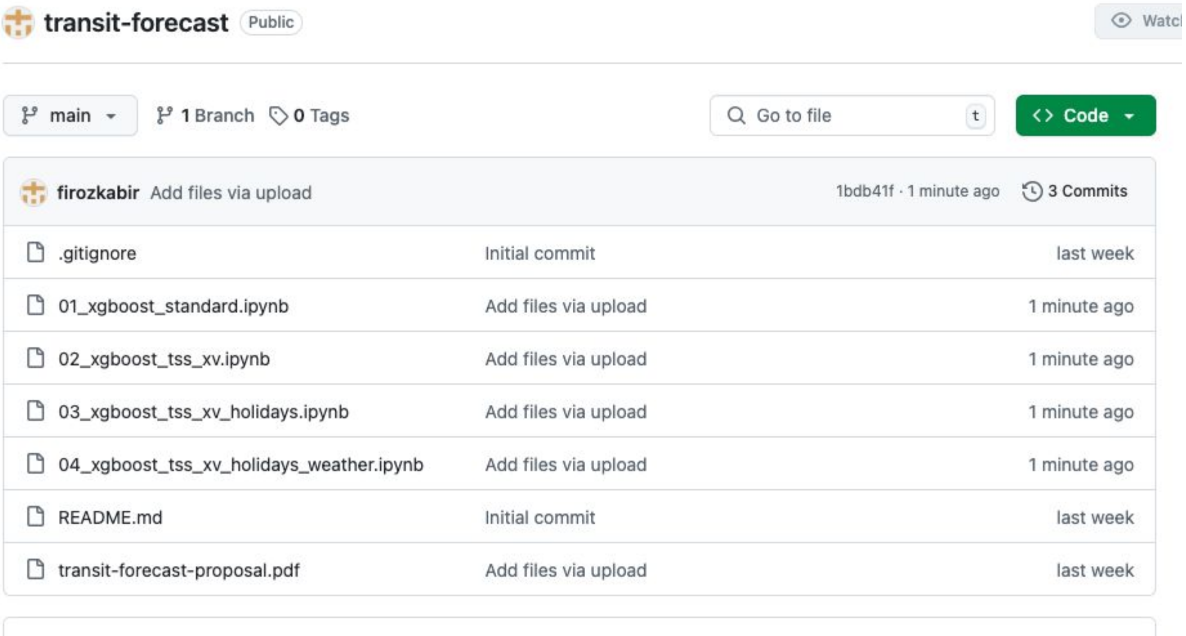
- ~~Problem Statement~~
- ~~Data Collection~~
- ~~EDA~~
- ~~Naive Mode~~
- ~~Additional Data~~
- ~~TimeSeris Split, Lag Features and Cross Validation~~
- ~~Enhanced Model~~
- ~~Model Evaluation~~
- **Conclusions**

Conclusions

- Outlier Detection
- Weather Data, Holiday Indicator
- TimeSeries Split
- Lag Features
- Cross Validation
- XGBoost

Source Code

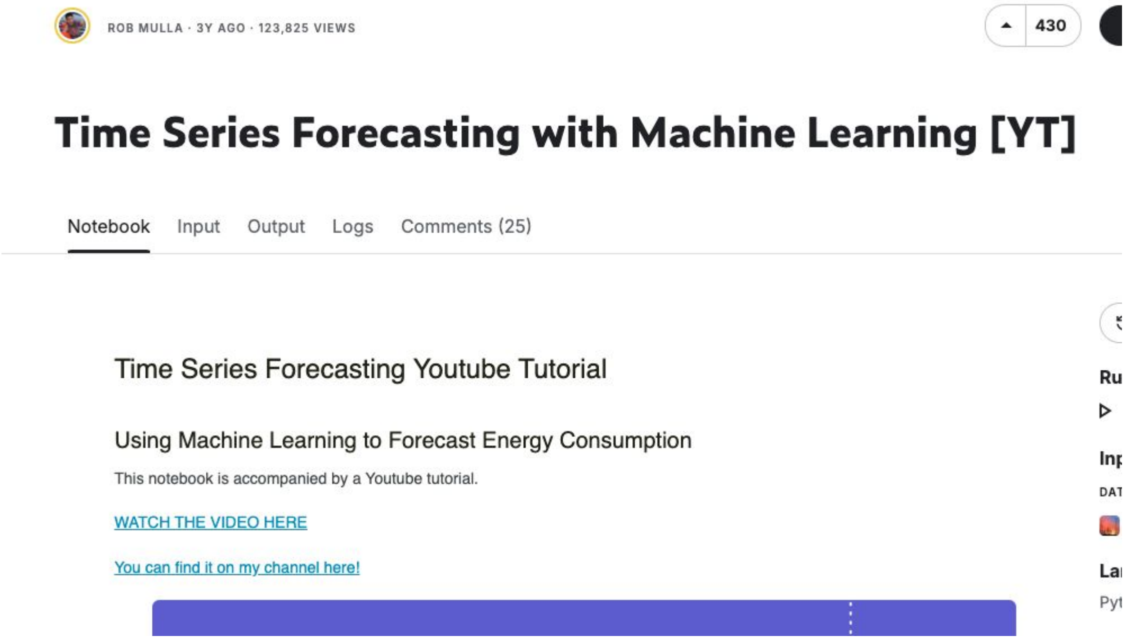
https://github.com/firozkabir/transit-forecast



Credits

https://www.kaggle.com/code/robikscube/time-series-forecasting-with-machine-learning-yt

https://www.kaggle.com/code/robikscube/tutorial-time-series-forecasting-with-xgboost



Questions