

The background of the slide is a photograph of a city skyline at sunset. On the left, several high-voltage electrical transmission towers are visible, with power lines stretching across the frame. On the right, modern skyscrapers are silhouetted against the bright, orange and yellow sky. The sun is low on the horizon, creating a strong lens flare effect. A teal-colored rectangular box is positioned in the upper center of the image, containing the text 'Toronto Hydro' in white. Below this box, the words 'Demand Prediction' are written in a large, bold, black font.

Toronto Hydro

Demand Prediction

Data Sources

<http://www.ieso.ca>

20 CSV file contain all hydro demanded from 2003 to 2020



Date	Hour	Ontario Demand	Northwest	Northeast	Ottawa	East	Toronto	Essa	Bruce	Southwest	Niagara	West	Zone	Total	Diff
2020-01-01	1	13219	523	1200	914	847	4718	902	64	2564	399	1290		13419	200
2020-01-01	2	12895	518	1190	887	826	4571	869	60	2504	384	1273		13082	187
2020-01-01	3	12554	519	1201	865	803	4443	839	60	2404	373	1260		12768	214
2020-01-01	4	12360	519	1183	852	789	4356	825	59	2406	361	1241		12591	231

<https://openweathermap.org>

The temperature of Toronto including all features in same period of time.



id	date	hour	dt	timezone	temp	feels_like	temp_min	temp_max	pressure	humidity	wind_speed	wind_deg	clouds_all	weather_main	weather_description
220537	2003-05-01	1	1051750800	-14400	7.52	1.21	3.32	9	1015	49	5.7	80	90	Clouds	overcast clouds
220538	2003-05-01	2	1051754400	-14400	7.03	1.53	3.52	8	1015	52	4.6	50	90	Rain	moderate rain
220539	2003-05-01	3	1051758000	-14400	6.57	-0.39	3.63	7.7	1014	65	7.2	50	90	Rain	light rain
220540	2003-05-01	4	1051761600	-14400	6.8	-1.47	3.65	8	1013	56	8.7	70	90	Clouds	overcast clouds

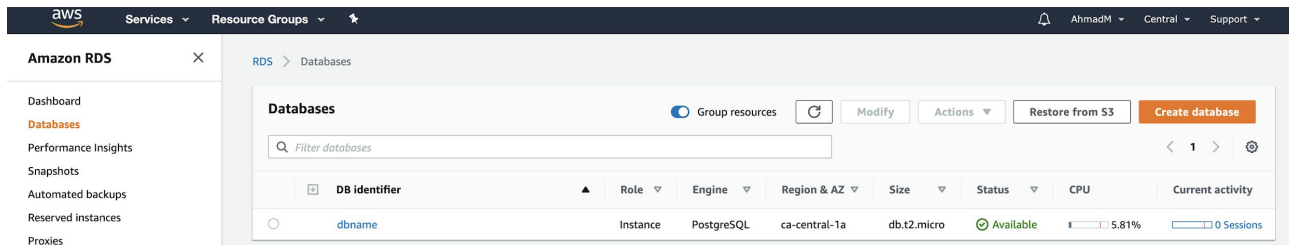
Data Structure

- Merging all CSVs from hydro to a single file in Python
- Cleaning all dataset such as converting UTC to GMT, splitting columns, etc.
- Uploading final data sources on to the AWS as a PostgreSQL.

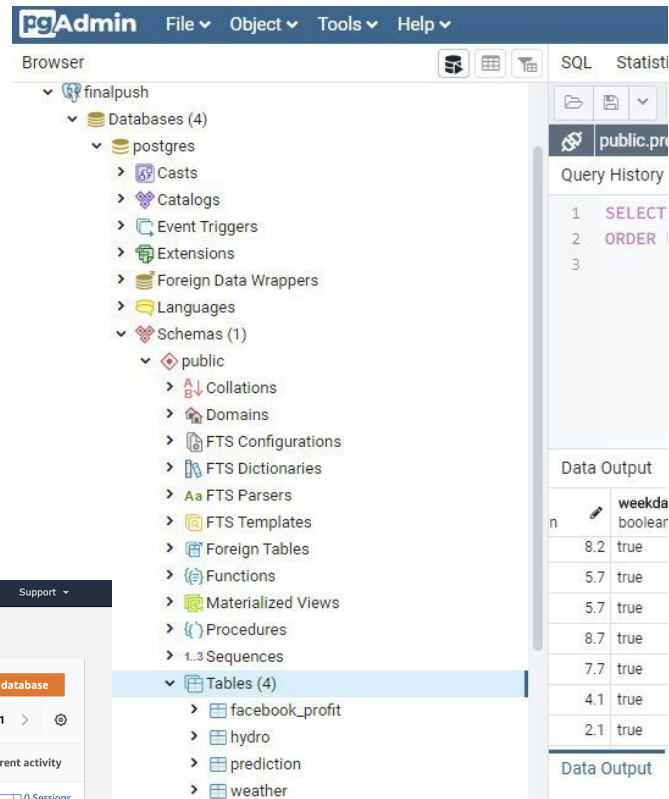
```
[1]: import os
import glob
import pandas as pd
os.chdir("Resources")
```

```
[2]: extension = 'csv'
all_filenames = [i for i in glob.glob('*.{}'.format(extension))]
```

```
[3]: combined_csv = pd.concat([pd.read_csv(f, skiprows=3) for f in all_filenames ])
combined_csv.to_csv( "combined_csv.csv", index=False, encoding='utf-8-sig')
```

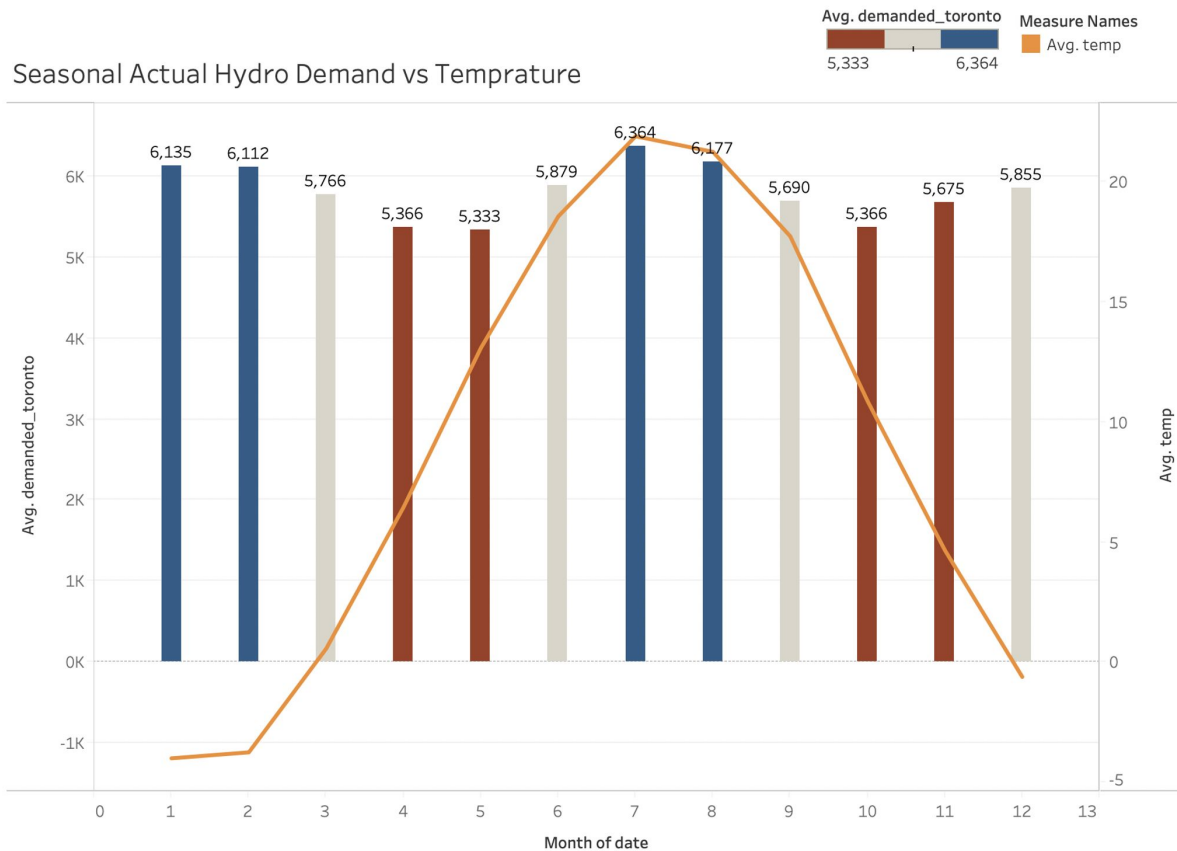


The screenshot shows the Amazon RDS console interface. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and a user profile 'AhmadM' with a dropdown menu. The left sidebar contains a navigation menu with 'Dashboard', 'Databases' (highlighted), 'Performance Insights', 'Snapshots', 'Automated backups', 'Reserved instances', and 'Proxies'. The main content area is titled 'Databases' and includes a search bar 'Filter databases'. Below the search bar is a table with columns: DB identifier, Role, Engine, Region & AZ, Size, Status, CPU, and Current activity. One database is listed: 'dbname' with role 'Instance', engine 'PostgreSQL', region 'ca-central-1a', size 'db.t2.micro', status 'Available', CPU usage '5.81%', and '0 Sessions'.



The screenshot shows the pgAdmin 4 web interface. The top menu bar includes 'pgAdmin', 'File', 'Object', 'Tools', and 'Help'. The 'Browser' pane on the left shows a tree view of the database structure: 'finalpush' > 'Databases (4)' > 'postgres' > 'Schemas (1)' > 'public'. The 'public' schema is expanded, showing various database objects like 'Collations', 'Domains', 'FTS Configurations', 'FTS Dictionaries', 'FTS Parsers', 'FTS Templates', 'Foreign Tables', 'Functions', 'Materialized Views', 'Procedures', 'Sequences', and 'Tables (4)'. The 'Tables (4)' sub-item is selected and expanded, showing a list of tables: 'facebook_profit', 'hydro', 'prediction', and 'weather'. On the right, the 'Query History' pane shows a list of queries, and the 'Data Output' pane shows a table with columns 'n', 'weekda', and 'boolean', containing data for the 'hydro' table.

Basic Analysis



Facebook Prophet- Time Series

PROPHET

```
from fbprophet import Prophet
from datetime import datetime
from fbprophet.plot import plot_plotly
import plotly.offline as py
```

► ML

```
df = pd.read_csv('combined_csv2.csv')
df.head(2)
```

	Date	Hour	Time	Ontario	Demand	Northwest	Northeast	Ottawa
0	2003-05-01	1	2003-05-01 00:59:59	13702	809	1284	96	
1	2003-05-01	2	2003-05-01 01:59:59	13578	825	1283	92	

► ML

```
new_df=df[['Time','Toronto']]
final_df = new_df.rename({'Time': 'ds', 'Toronto': 'y'}, axis=1)
final_df['Time'] = pd.to_datetime(df['Time'])
#sample=final_df.tail(26288)
final_df.head(2)
#sample.tail()
```

	ds	y	Time
0	2003-05-01 00:59:59	4422	2003-05-01 00:59:59
1	2003-05-01 01:59:59	4340	2003-05-01 01:59:59

► ML

```
m = Prophet(yearly_seasonality=True)
m.fit(final_df)
```

<fbprophet.forecaster.Prophet at 0x132dae310>

► ML

```
#future=pd.read_csv('predict.csv')
future = m.make_future_dataframe(freq="H",periods=72)
future.tail(2)
```

	ds
150838	2020-07-14 22:59:59
150839	2020-07-14 23:59:59

► ML

```
forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail(2)
```

\hat{y}

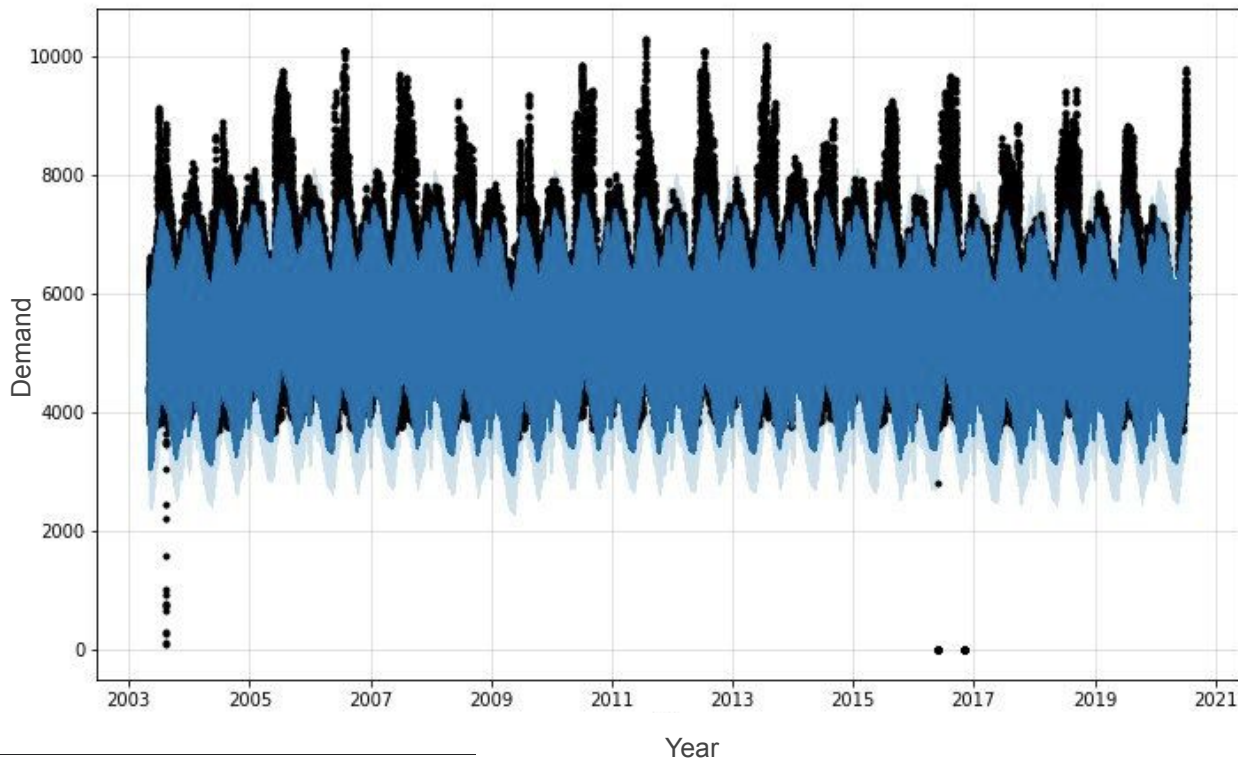
Upper

\hat{y}

Prediction

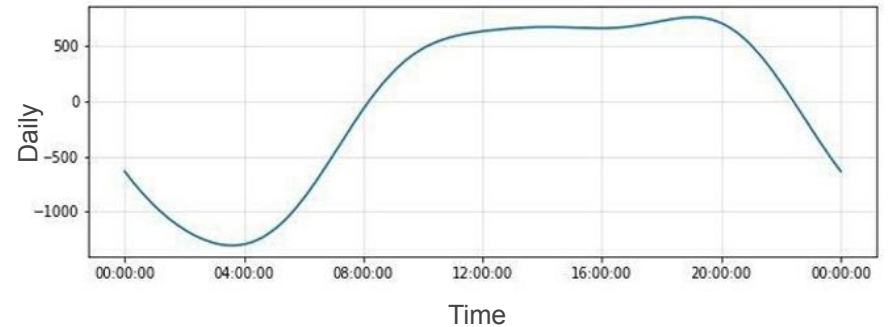
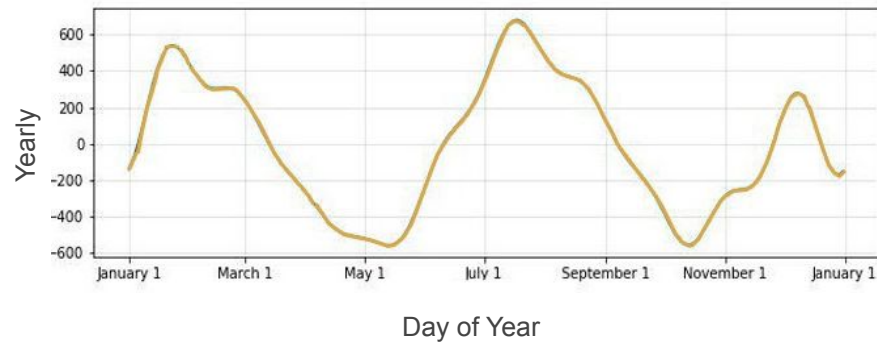
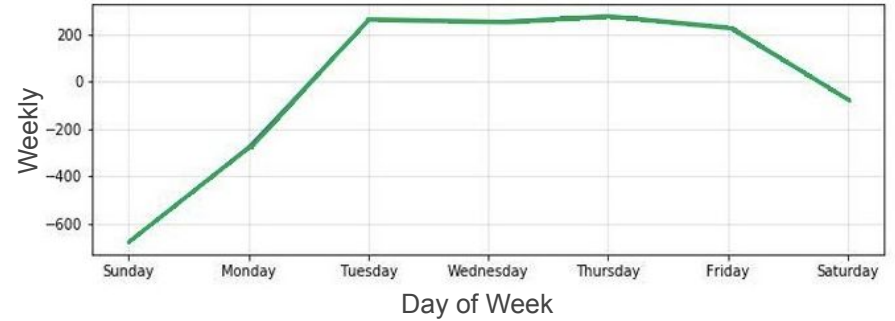
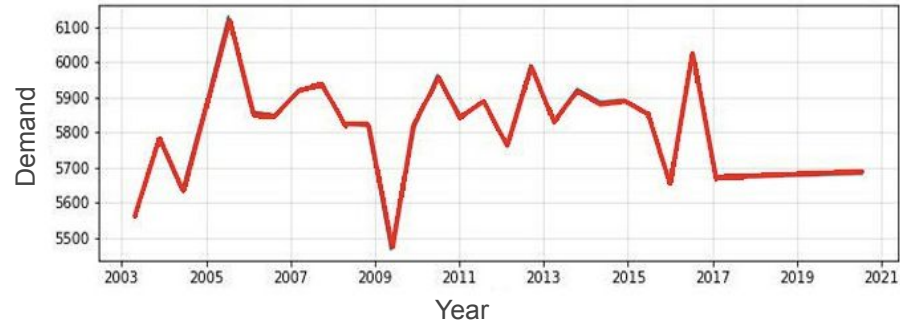
\hat{y}

Lower



Facebook Prophet- Time Series

PROPHET



Machine Learning- Scikit Linear Regression



```
Linear_Regression_Model.ipynb
File Edit View Insert Runtime Tools Help

+ Code + Text

[ ] import pandas as pd
    %matplotlib inline
    import matplotlib.pyplot as plt
    import numpy as np
    from sqlalchemy import create_engine
    import sqlalchemy
    from sqlalchemy.ext.automap import automap_base
    from sqlalchemy.orm import Session
    from sqlalchemy import create_engine, func
    import datetime

HYDRO DATA

# Reading Hydro Data
engine = create_engine('postgresql://postgres:postgres@dbname.cwx2xnixkpb1.ca-central-1.rds.amazonaws.com/postgres')

# Reflect an existing database into a new model
Base = automap_base()

# Reflect the tables
Base.prepare(engine, reflect=True)

# Save reference to the table
hydro = Base.classes.hydro
weather = Base.classes.weather

[ ] session = Session(engine)
    results = session.query(hydro.date,hydro.hour,hydro.demanded_toronto,hydro.weekday, hydro.previous_hour_demand, hydro.

    session.close()

    hydro = []
    for date, hour, demanded_toronto, weekday, previous_hour_demand, previous_day_demand in results:
        hydro_dict = {}
        hydro_dict["date"] = date
        hydro_dict["hour"] = hour
        hydro_dict["demanded_toronto"] = demanded_toronto
        hydro_dict["weekday"] = weekday
        hydro_dict["previous_hour_demand"] = previous_hour_demand
        hydro_dict["previous_day_demand"] = previous_day_demand

        hydro.append(hydro_dict)
```

```
Linear_Regression_Model.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[ ] from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, random

[ ] from sklearn.linear_model import LinearRegression
    model = LinearRegression()
    model.fit(X_train, y_train)

LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, n

[ ] X_train

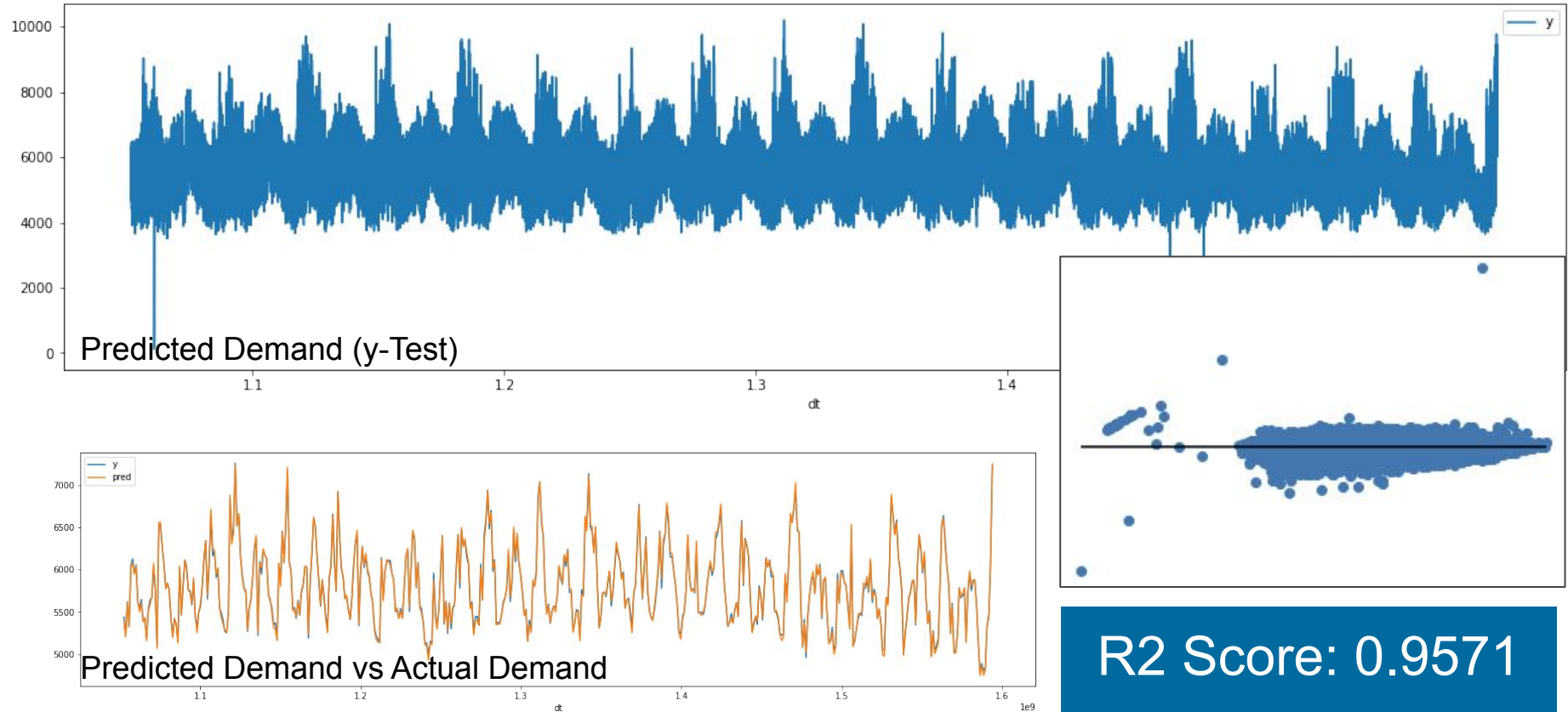
[ ] score = model.score(X_train, y_train)
    print(f"R2 Score: {score}")

R2 Score: 0.9574600025557655

[ ] score = model.score(X_test, y_test)
    print(f"R2 Score: {score}")

R2 Score: 0.9570905237024288
```

Machine Learning– Scikit Linear Regression



R2 Score: 0.9571

Machine Learning- Neural Network

```
Neural_Network_Model.ipynb
File Edit View Insert Runtime Tools Help Last saved at 10:51 PM

+ Code + Text

[ ] # Function to create the keras model

def build_model():
    model = keras.Sequential([
        layers.Dense(units=200, activation='relu', input_dim=X_train2.shape[1]),
        layers.Dense(units=200, activation='relu'),
        layers.Dense(units=1)
    ])

    optimizer = tf.keras.optimizers.RMSprop(0.001)

    model.compile(loss='mse',
                  optimizer=optimizer,
                  metrics=['mae', 'mse'])

    return model

[ ] # Create a model object
model = build_model()

[ ] # Check the model
model.summary()

Model: "sequential"
Layer (type) Output Shape Param #
=====
dense (Dense) (None, 200) 2400
dense_1 (Dense) (None, 200) 40200
dense_2 (Dense) (None, 1) 201
=====
Total params: 42,801
Trainable params: 42,801
Non-trainable params: 0

[ ] # save the model
model.save(' /content/gdrive/My Drive/Final Project/HydroDemand_NeuralModel')
```

INFO:tensorflow:Assets written to: /content/gdrive/My Drive/Final Project/HydroDemand_NeuralModel/assets

```
Neural_Network_Model.ipynb
File Edit View Insert Runtime Tools Help Last saved at 10:51 PM

+ Code + Text

[ ] # use model to predict using testing data
test_predictions = model.predict(X_test2).flatten()

a = plt.axes(aspect='equal')
plt.scatter(y_test, test_predictions)
plt.xlabel('True Values [Demand]')
plt.ylabel('Predictions [Demand]')
lims = [0, max(y_test)]
plt.xlim(lims)
plt.ylim(lims)
_ = plt.plot(lims, lims)

[ ] plt.savefig('/content/drive/My Drive/Final Project/Neural_Prediction_Xtest.jpg')

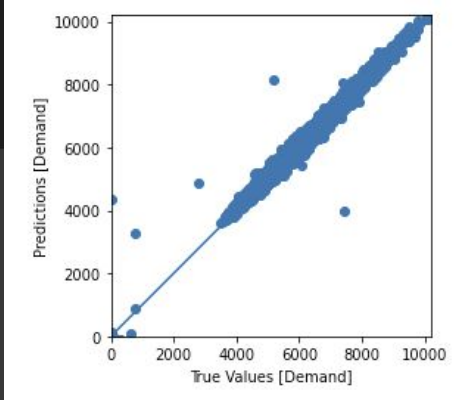
<Figure size 432x288 with 0 Axes>

[ ] # model = keras.models.load_model('/content/drive/My Drive/Final Project/HydroDemand_NeuralModel/')

[ ] pred = model.predict(X_test2)

[ ] r2_score(y_test, pred)

0.9879894579676439
```

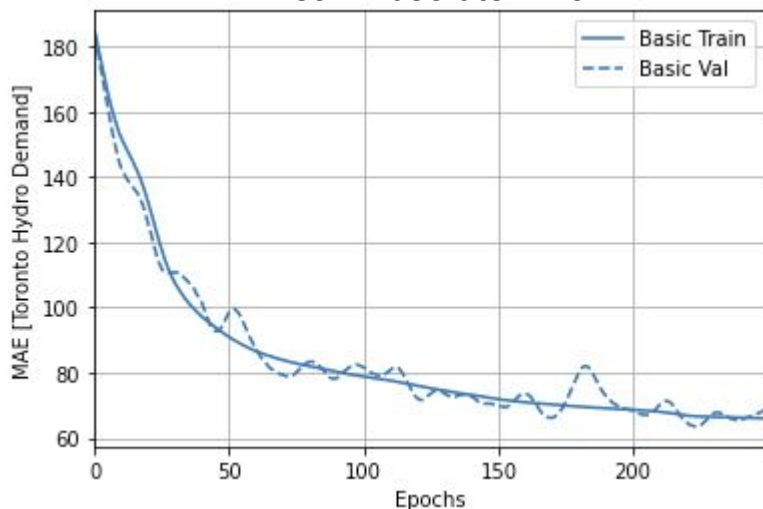


Machine Learning- Neural Network

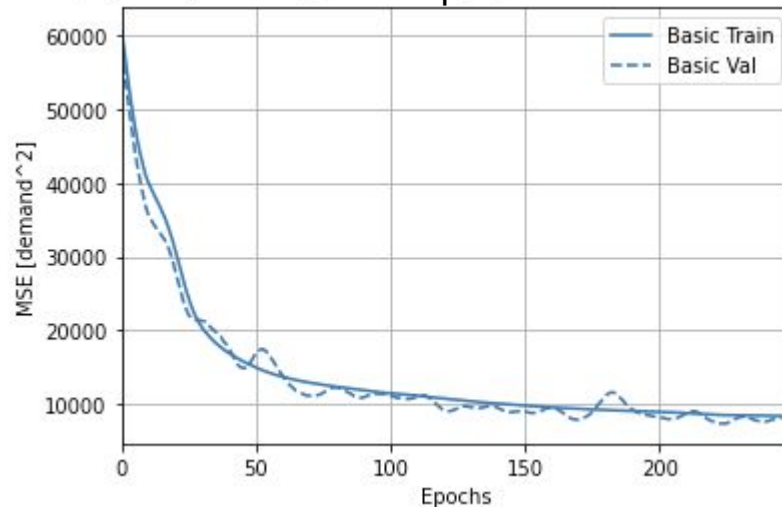


TensorFlow

Mean Absolute Error



Mean Square Error



R2 Score: 0.9880

	loss	mae	mse	val_loss	val_mae	val_mse	epoch
245	8437.715820	65.906090	8437.715820	13518.171875	86.379005	13518.171875	245
246	8421.664062	66.187973	8421.664062	6475.656738	59.384483	6475.656738	246
247	8441.654297	66.013245	8441.654297	5915.229492	56.081337	5915.229492	247
248	8386.368164	65.855492	8386.368164	6543.145996	58.717964	6543.145996	248
249	8341.509766	65.975052	8341.509766	11230.646484	89.272881	11230.646484	249

Tableau- Final Analysis



Prediction vs Actual Data (2003-2020)
Linear Regression Model

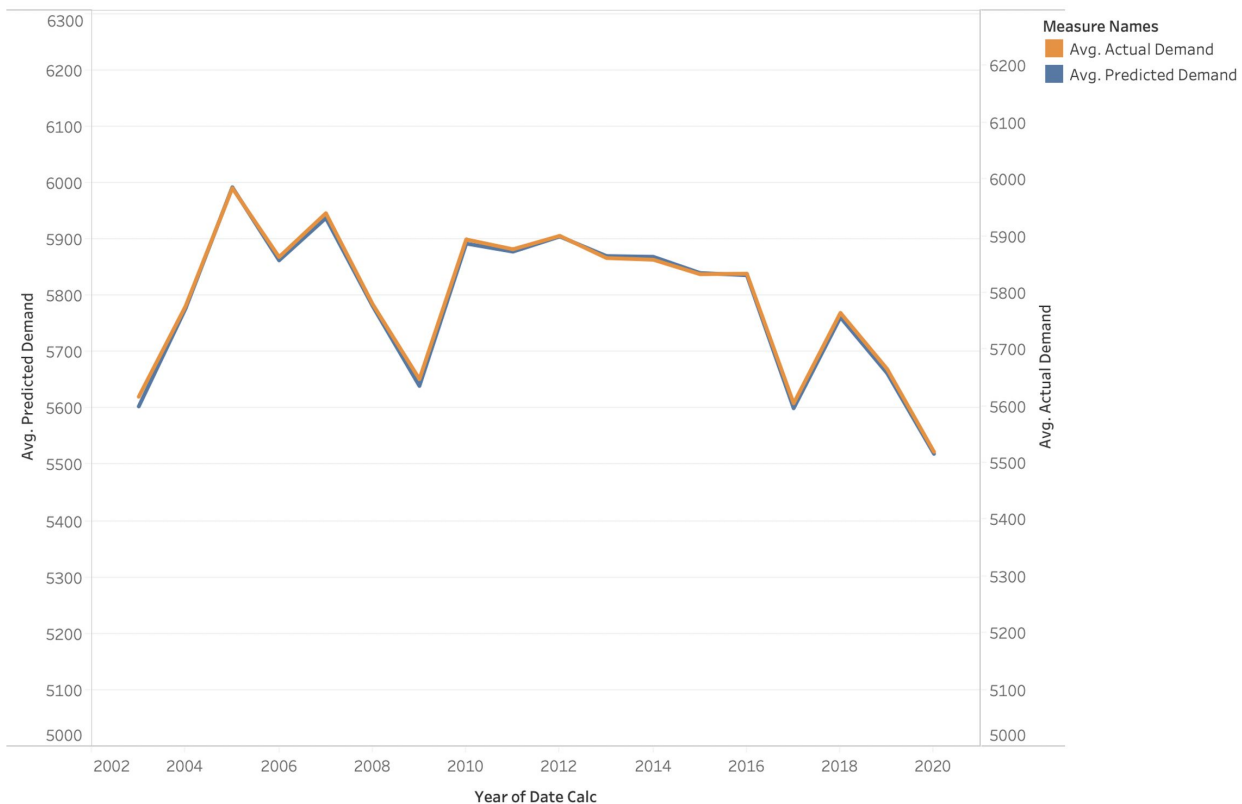


Tableau- Final Analysis



Prediction vs Actual Data (2019-2020)
Linear Regression Model

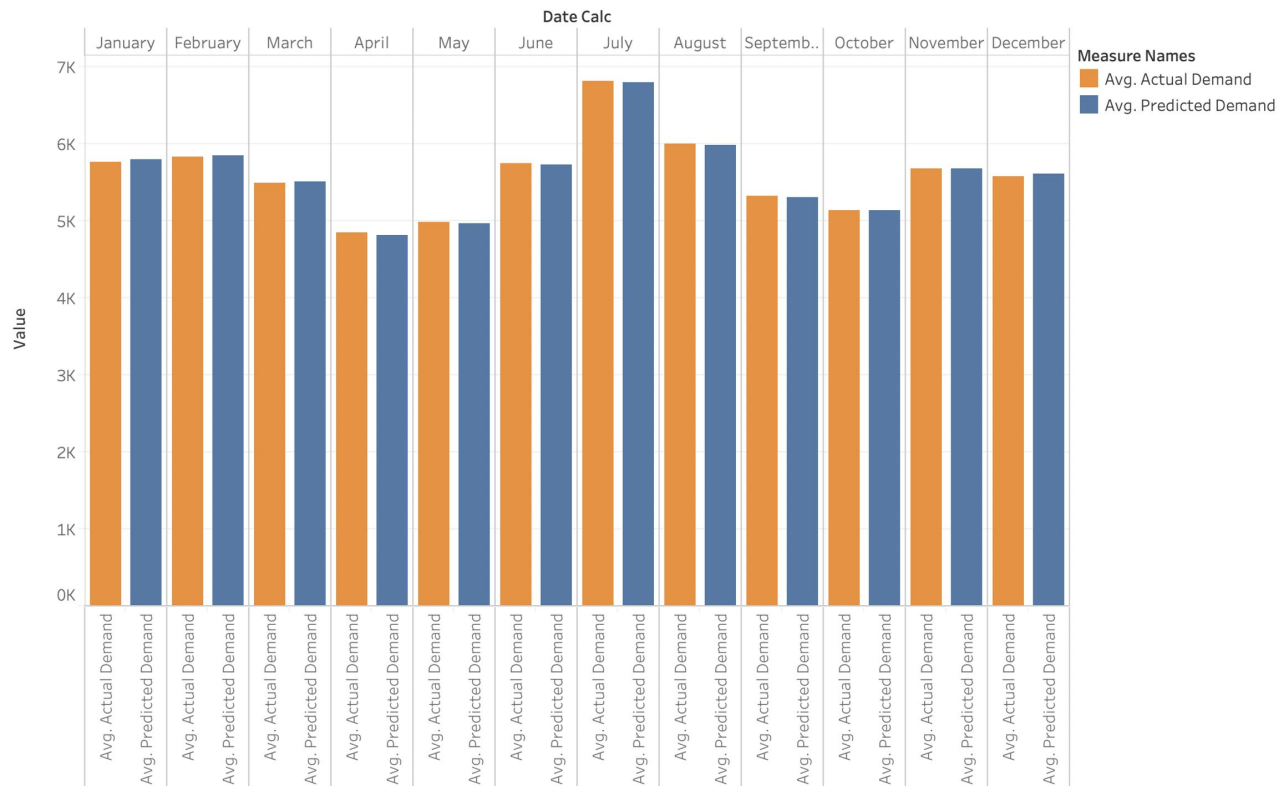


Tableau- Final Analysis



Prediction vs Actual Data (2020, June 11- July 11)
Linear Regression Model

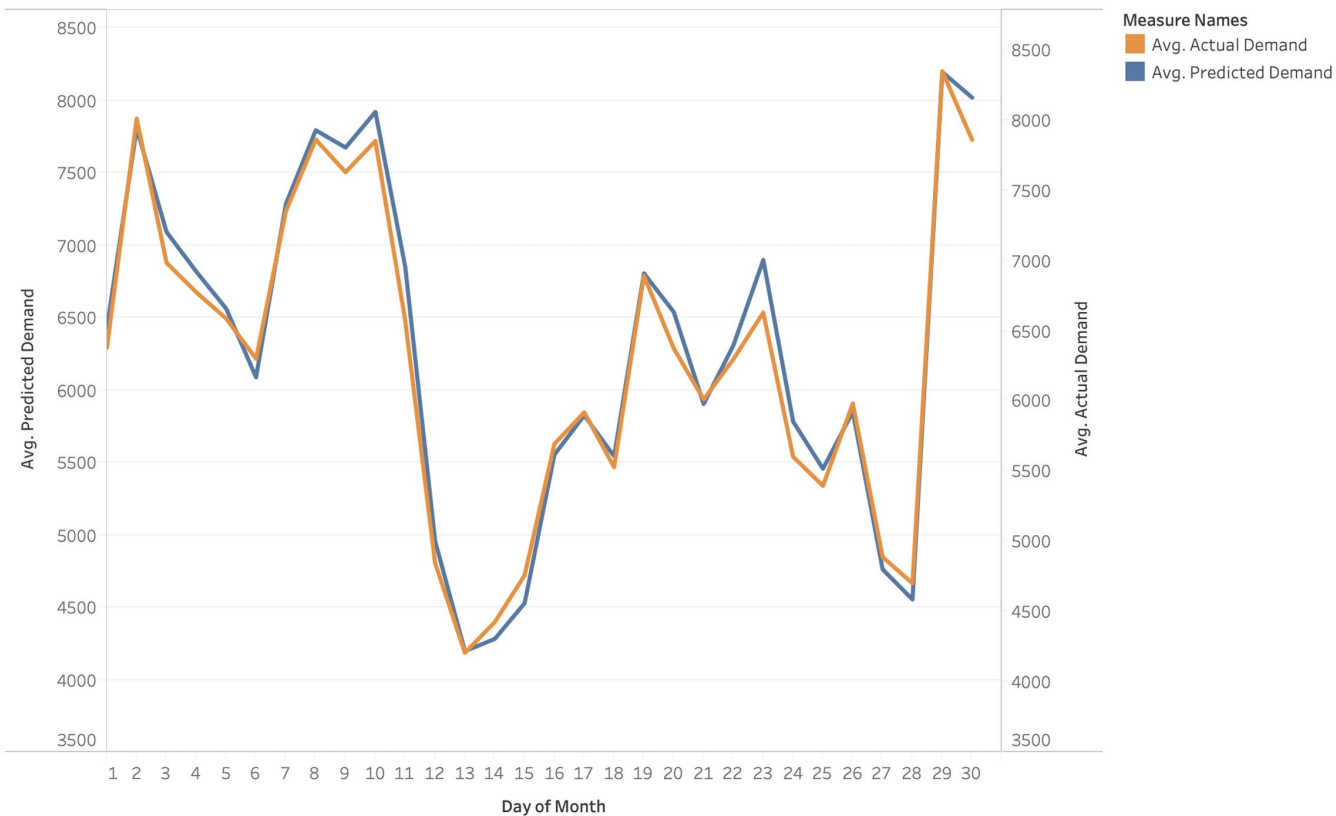


Tableau- Final Analysis



Prediction vs Actual Data (Hours of One Day)
Linear Regression Model

