CAPSTONE PROJECT 106

Sin Yi, Evan

PROBLEM STATEMENT

Company A would like to make an investment in Seoul. They oversee that there is a potential market in bike rental market as the market nowadays tends to green tech.

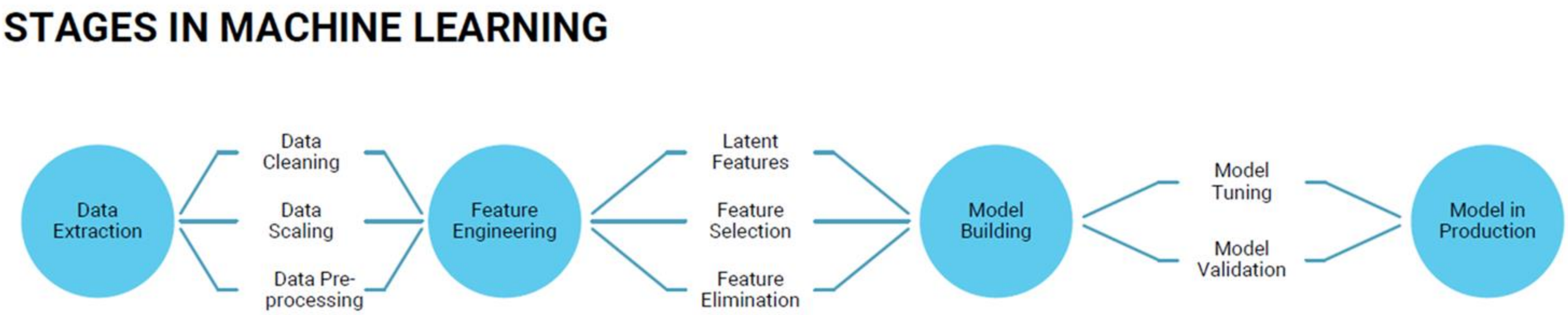
Company A approached us to do market survey on how’s the market in Seoul.

OBJECTIVE

1. To see the market trend of bike rental in Seoul.
2. To predict the bike count in terms of season and hours to have stable supply of rental bikes.

PROJECT FLOWCHART

1. Data Selection & Cleaning
2. Data Filtering & Preprocessing
3. EDA
4. Feature Engineering
5. Feature Selection
6. Model Building
7. Model Training and Testing
8. Model Evaluation & Hyper Parameter tuning
9. Model Deployment



Dataset - Seoul Bike Sharing Demand

Source: https://archive.ics.uci.edu/ml/datasets/Seoul+Bike+Sharing+Demand

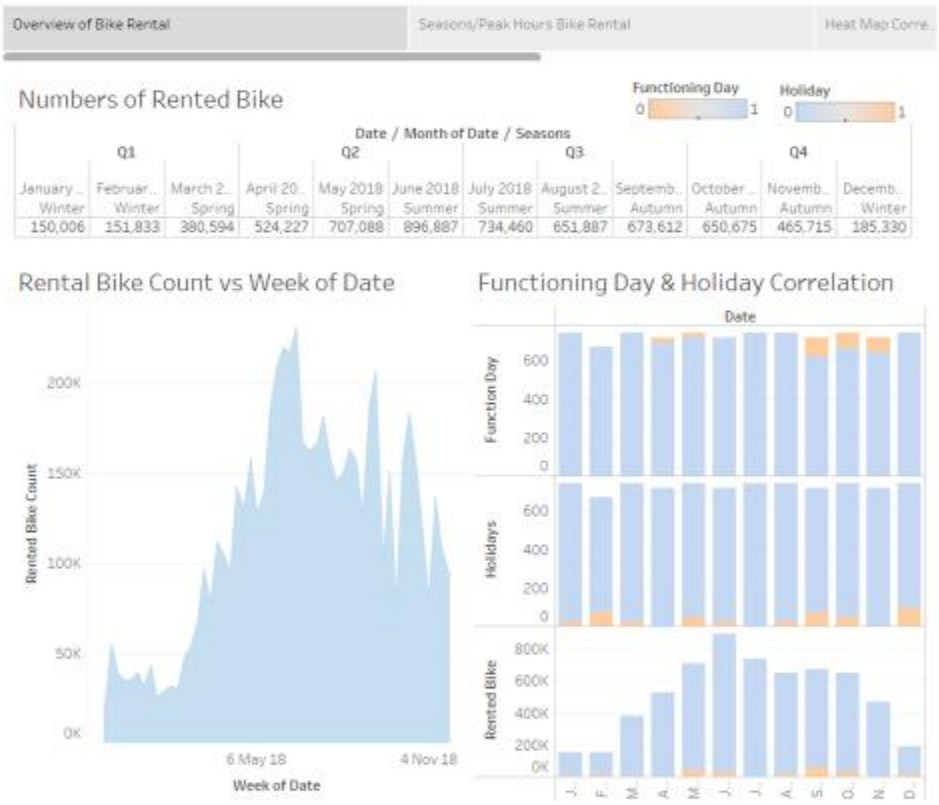
Date	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	Seasons	Holiday	Functioning Day
1/12/2017	254	0	-5.2	37	2.2	2000	-17.6	0	0	0	Winter	No Holiday	Yes
1/12/2017	204	1	-5.5	38	0.8	2000	-17.6	0	0	0	Winter	No Holiday	Yes
1/12/2017	173	2	-6	39	1	2000	-17.7	0	0	0	Winter	No Holiday	Yes
1/12/2017	107	3	-6.2	40	0.9	2000	-17.6	0	0	0	Winter	No Holiday	Yes
1/12/2017	78	4	-6	36	2.3	2000	-18.6	0	0	0	Winter	No Holiday	Yes
1/12/2017	100	5	-6.4	37	1.5	2000	-18.7	0	0	0	Winter	No Holiday	Yes
1/12/2017	181	6	-6.6	35	1.3	2000	-19.5	0	0	0	Winter	No Holiday	Yes
1/12/2017	460	7	-7.4	38	0.9	2000	-19.3	0	0	0	Winter	No Holiday	Yes
1/12/2017	930	8	-7.6	37	1.1	2000	-19.8	0.01	0	0	Winter	No Holiday	Yes
1/12/2017	490	9	-6.5	27	0.5	1928	-22.4	0.23	0	0	Winter	No Holiday	Yes
1/12/2017	339	10	-3.5	24	1.2	1996	-21.2	0.65	0	0	Winter	No Holiday	Yes
1/12/2017	360	11	-0.5	21	1.3	1936	-20.2	0.94	0	0	Winter	No Holiday	Yes
1/12/2017	449	12	1.7	23	1.4	2000	-17.2	1.11	0	0	Winter	No Holiday	Yes
1/12/2017	451	13	2.4	25	1.6	2000	-15.6	1.16	0	0	Winter	No Holiday	Yes
1/12/2017	447	14	3	26	2	2000	-14.6	1.01	0	0	Winter	No Holiday	Yes
1/12/2017	463	15	2.1	36	3.2	2000	-11.4	0.54	0	0	Winter	No Holiday	Yes
1/12/2017	484	16	1.2	54	4.2	793	-7	0.24	0	0	Winter	No Holiday	Yes
1/12/2017	555	17	0.8	58	1.6	2000	-6.5	0.08	0	0	Winter	No Holiday	Yes

Data Features

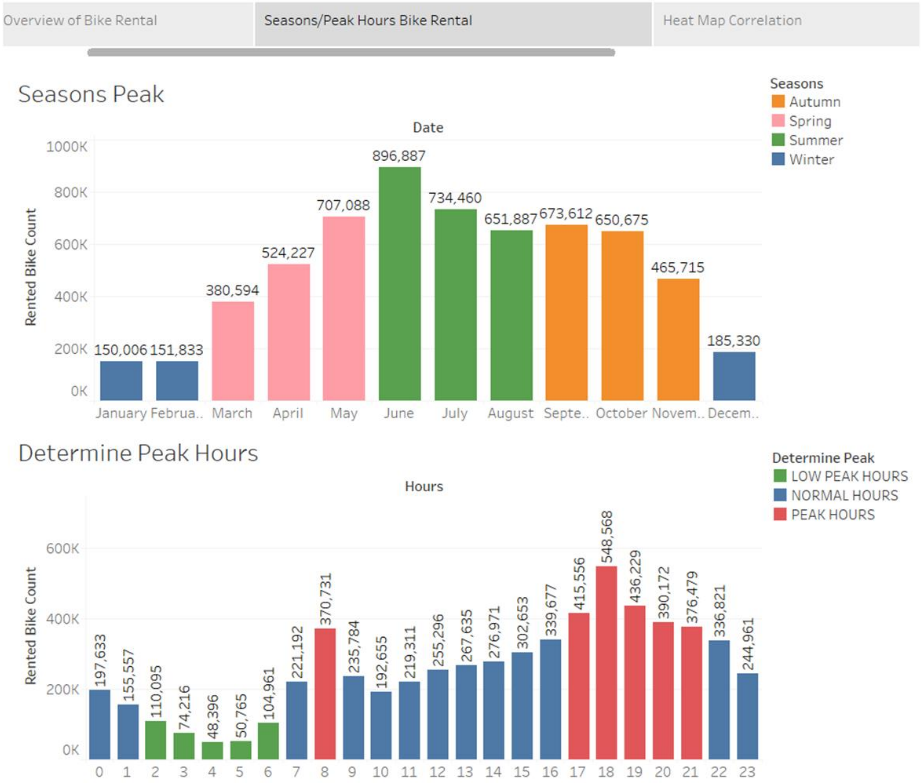
The dataset contains 8760 instances and 14 attributes.

Attributes	Description
Date	Date of Rented Bike
Rented Bike count	Number of total rentals
Hour	Hours of the day
Temperature(°C)	Weather Temperature in °C
Humidity(%)	Humidity of the day in %
Wind speed (m/s)	Wind speed in m/s
Visibility (10m)	Atmospheric Visibility within 10 m range
Dew point temperature(°C)	Dew Point Temperature - T dp in °C
Solar Radiation (MJ/m2)	Indicate light and energy that comes from the sun in MJ/m2
Rainfall(mm)	Rainfall in mm
Snowfall (cm)	Snowfall in cm
Seasons	Autumn, Spring, Summer, Winter
Holiday	Whether the day is considered a holiday
Functioning Day	Whether the day is function for bike rental

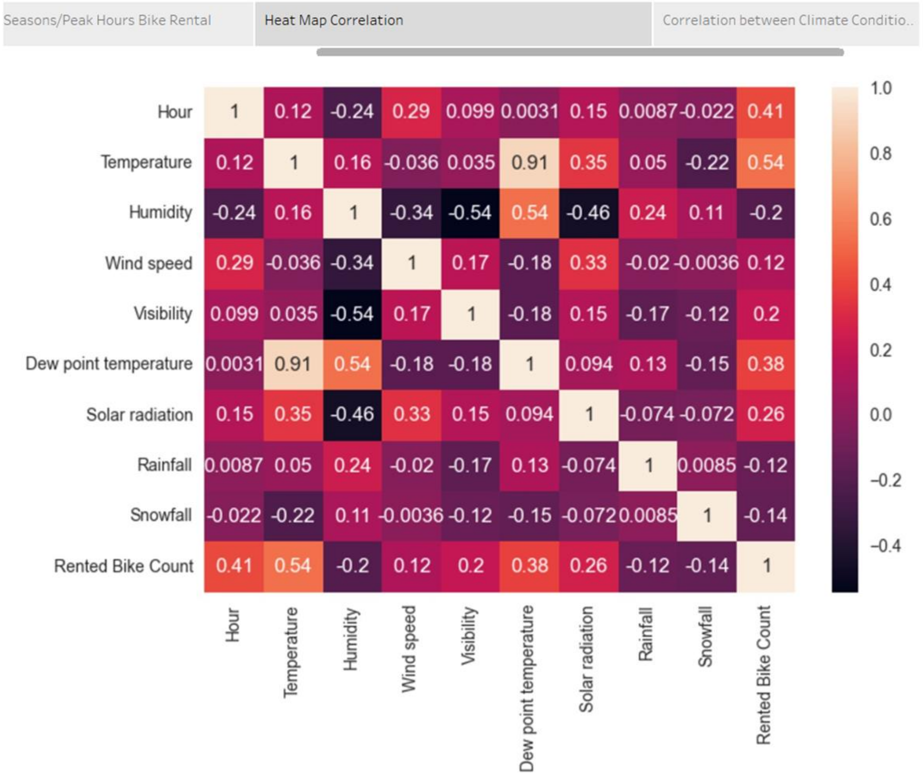
Exploratory Data Analysis (EDA)



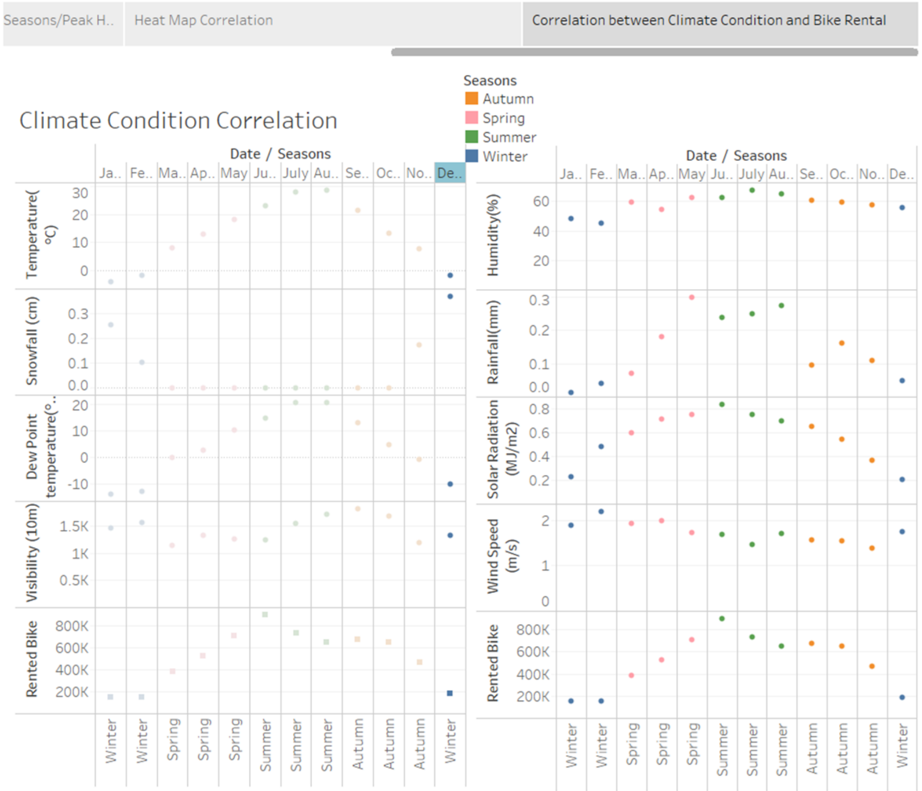
Exploratory Data Analysis (EDA)



Exploratory Data Analysis (EDA)



Exploratory Data Analysis (EDA)



FEATURE ENGINEERING

Label Encoding

- Convert Season, Holiday & Functioning Day from string to integer for modeling purpose

Date	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	Seasons	Holiday	Functioning Day
1/12/2017	254	0	-5.2	37	2.2	2000	-17.6	0	0	0	Winter	No Holiday	Yes
1/12/2017	204	1	-5.5	38	0.8	2000	-17.6	0	0	0	Winter	No Holiday	Yes
1/12/2017	173	2	-6	39	1	2000	-17.7	0	0	0	Winter	No Holiday	Yes
1/12/2017	107	3	-6.2	40	0.9	2000	-17.6	0	0	0	Winter	No Holiday	Yes
1/12/2017	78	4	-6	36	2.3	2000	-18.6	0	0	0	Winter	No Holiday	Yes

Original Data

```
encoded_df = pd.get_dummies(df, columns = ["Seasons", "Holiday", "Functioning Day"], drop_first = True)
encoded_df
```

	Date	Rented Bike Count	Hour	Temperature	Humidity	Wind speed	Visibility	Dew point temperature	Solar radiation	Rainfall	Snowfall	Seasons_Spring	Seasons_Summer	Seasons_Winter	Holiday_No Holiday	Functioning Day_Yes
0	01/12/2017	254	0	-5.2	37	2.2	2000	-17.6	0.0	0.0	0.0	0	0	1	1	1
1	01/12/2017	204	1	-5.5	38	0.8	2000	-17.6	0.0	0.0	0.0	0	0	1	1	1
2	01/12/2017	173	2	-6.0	39	1.0	2000	-17.7	0.0	0.0	0.0	0	0	1	1	1
3	01/12/2017	107	3	-6.2	40	0.9	2000	-17.6	0.0	0.0	0.0	0	0	1	1	1
4	01/12/2017	78	4	-6.0	36	2.3	2000	-18.6	0.0	0.0	0.0	0	0	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8755	30/11/2018	1003	19	4.2	34	2.6	1894	-10.3	0.0	0.0	0.0	0	0	0	1	1
8756	30/11/2018	764	20	3.4	37	2.3	2000	-9.9	0.0	0.0	0.0	0	0	0	1	1

After Encoding

Regression model - Using PyCaret

Setting up the PyCaret environment

```
reg = setup(data = encoded_df,
            target = 'Rented Bike Count',
            numeric_imputation = 'mean',
            normalize = True)
```

	Description	Value
0	Session id	4804
1	Target	Rented Bike Count
2	Target type	Regression
3	Data shape	(8760, 19)
4	Train data shape	(6132, 19)
5	Test data shape	(2628, 19)
6	Numeric features	14
7	Date features	1
8	Categorical features	1
9	Preprocess	True
10	Imputation type	simple
11	Numeric imputation	mean
12	Categorical imputation	mode
13	Maximum one-hot encoding	25
14	Encoding method	None
15	Normalize	True
16	Normalize method	zscore
17	Fold Generator	KFold
18	Fold Number	10
19	CPU Jobs	-1
20	Use GPU	False
21	Log Experiment	False
22	Experiment Name	reg-default-name
23	USI	c9d9

70% used as train dataset

Compare models

Top 2 Model:

1. lightgbm - Light Gradient Boosting Machine
2. Catboost -CatBoost Regressor

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
lightgbm	Light Gradient Boosting Machine	142.0291	49937.2518	223.3045	0.8783	0.9068	0.5230	0.3580
catboost	CatBoost Regressor	142.2654	50046.9535	223.4763	0.8780	0.9359	0.5235	2.7860
et	Extra Trees Regressor	139.7847	52395.2809	228.6487	0.8724	0.5684	0.5391	0.8450
xgboost	Extreme Gradient Boosting	149.3723	54495.4810	233.1026	0.8671	0.9530	0.5581	0.5350
rf	Random Forest Regressor	144.8364	54958.8450	234.2672	0.8661	0.6779	0.5626	1.2360
gbr	Gradient Boosting Regressor	172.2491	66572.6993	257.8067	0.8380	1.0304	0.7494	0.4810
knn	K Neighbors Regressor	195.9716	94128.7385	306.3947	0.7706	0.6034	0.7421	0.0750
dt	Decision Tree Regressor	188.0859	100787.1353	317.1107	0.7546	0.6326	0.6138	0.0530
br	Bayesian Ridge	322.4843	188784.9295	434.3888	0.5402	1.3824	1.6645	0.0210
ridge	Ridge Regression	322.4902	188832.1578	434.4441	0.5401	1.3814	1.6618	0.0300
lr	Linear Regression	322.4901	188842.5615	434.4562	0.5401	1.3814	1.6615	1.7010
lar	Least Angle Regression	322.4901	188842.5615	434.4562	0.5401	1.3814	1.6615	0.0270
lasso	Lasso Regression	322.5126	188868.4218	434.4860	0.5400	1.3783	1.6650	0.0680
ada	AdaBoost Regressor	370.8939	189918.0052	435.5290	0.5372	1.5543	2.3825	0.3290
huber	Huber Regressor	313.7101	195269.3465	441.7853	0.5246	1.3478	1.4466	0.0370
par	Passive Aggressive Regressor	317.8825	202555.4662	449.9561	0.5068	1.3336	1.5109	0.0310
en	Elastic Net	331.7979	208572.8226	456.5811	0.4924	1.3559	1.7103	0.0250
llar	Lasso Least Angle Regression	344.7186	222263.7798	471.3160	0.4591	1.4342	2.0608	0.0270
omp	Orthogonal Matching Pursuit	402.7180	293899.3780	542.0435	0.2846	1.6153	2.4285	0.0210
dummy	Dummy Regressor	515.8574	411043.4262	641.0136	-0.0004	1.7282	3.3418	0.0320

Creating Model - Light Gradient Boosting Machine

Before Tuning

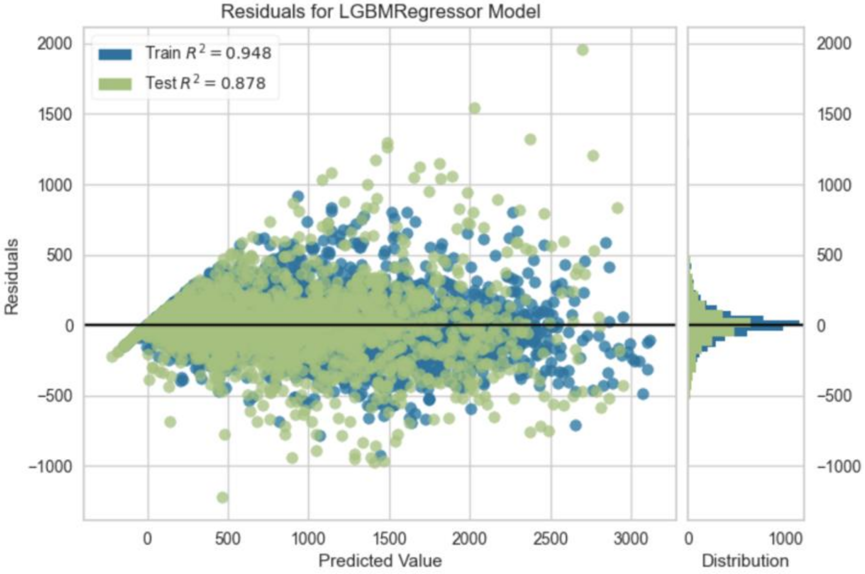
Fold	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	143.2532	53137.5615	230.5159	0.8624	0.9170	0.4072
1	137.5423	45427.7694	213.1379	0.8901	0.9328	0.5560
2	141.0592	49769.4118	223.0906	0.8808	0.8712	0.4763
3	148.5883	51456.5429	226.8403	0.8758	0.8269	0.6029
4	139.6704	46201.4542	214.9452	0.8876	0.8326	0.4833
5	149.8197	54741.1636	233.9683	0.8688	0.9063	0.6244
6	146.8227	56796.0821	238.3193	0.8631	0.9862	0.4450
7	137.6392	46702.3317	216.1072	0.8842	0.9964	0.6173
8	137.2604	45627.5020	213.6060	0.8823	0.9544	0.5780
9	138.6355	49512.6989	222.5145	0.8884	0.8446	0.4400
Mean	142.0291	49937.2518	223.3045	0.8783	0.9068	0.5230
Std	4.5646	3823.7420	8.5055	0.0098	0.0587	0.0773

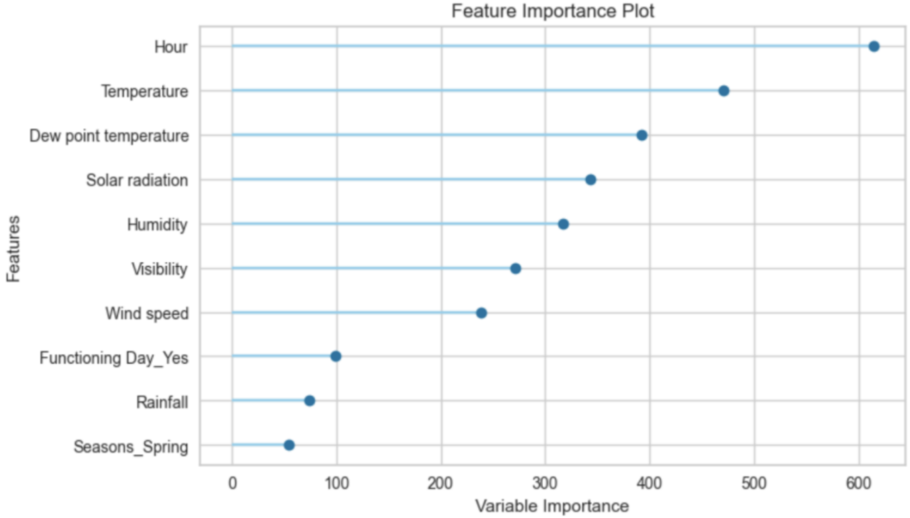
After tuning

	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	140.6701	50444.5304	224.5986	0.8833	1.0743	0.6342
1	156.5665	58681.4594	242.2426	0.8640	0.8800	0.5354
2	141.4736	52307.5743	228.7085	0.8746	0.7533	0.4329
3	146.4117	49176.7324	221.7583	0.8868	0.7969	0.4464
4	142.6814	51035.4682	225.9103	0.8744	0.9446	0.4472
5	144.5763	51872.0979	227.7545	0.8819	0.9721	0.5134
6	148.7100	58329.7947	241.5156	0.8602	0.9406	0.4918
7	142.2381	51126.9989	226.1128	0.8638	0.8009	0.5282
8	139.6006	53551.1160	231.4111	0.8683	1.0544	0.4244
9	141.0088	47370.9907	217.6488	0.8837	0.9635	0.7117
Mean	144.3937	52389.6763	228.7661	0.8741	0.9181	0.5165
SD	4.8509	3452.7837	7.4663	0.0091	0.1030	0.0885

LGBM

Residual Plot





- High density of points close to origin & low density of points away from the origin
- Symmetric about the origin

LGBM

- Predicting on test sample

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Light Gradient Boosting Machine	138.5363	47209.3378	217.2771	0.8858	0.9316	0.5035

- Finalising the model

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Light Gradient Boosting Machine	112.5486	29131.8041	170.6804	0.9295	0.8346	0.4237

Create model - CatBoost Regressor

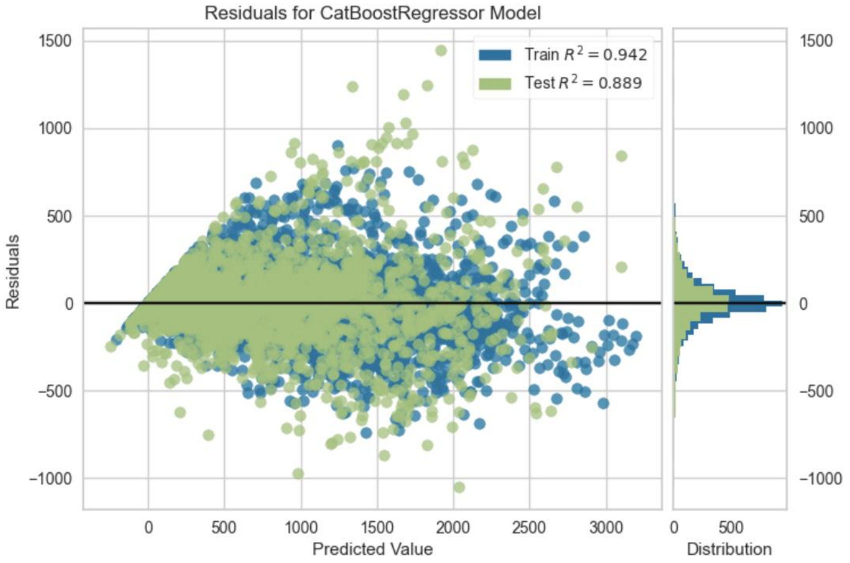
Before Tuning

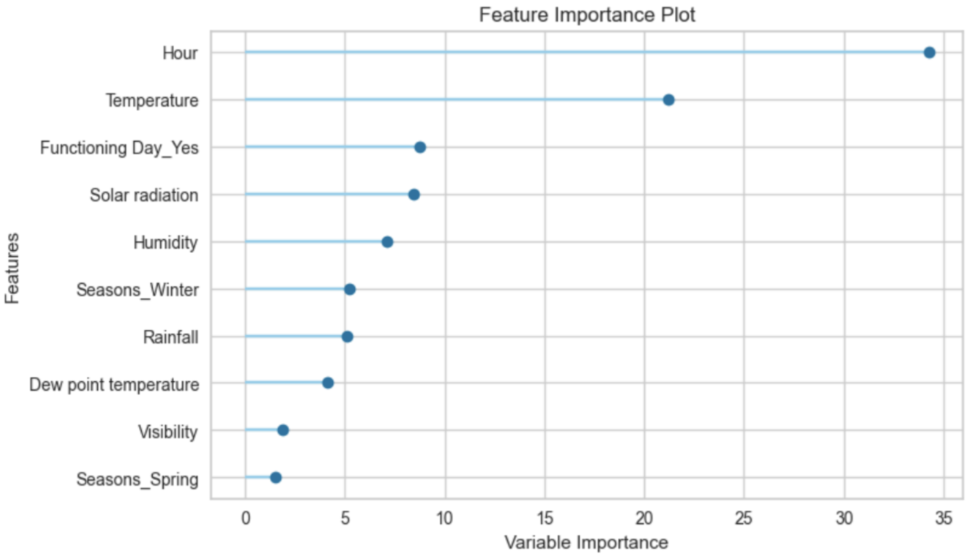
	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	141.3603	51531.8852	227.0064	0.8807	0.9459	0.6862
1	154.9105	59584.9150	244.1002	0.8619	0.8895	0.5553
2	139.2181	48825.5894	220.9651	0.8829	0.7772	0.4321
3	141.5873	46994.9893	216.7833	0.8918	0.8421	0.4109
4	143.4578	49341.6901	222.1299	0.8786	0.9140	0.4279
5	140.8262	49030.4216	221.4281	0.8884	0.9320	0.4795
6	146.2195	54363.3723	233.1595	0.8697	0.8998	0.4721
7	134.1486	41988.9640	204.9121	0.8881	0.7860	0.4844
8	141.5106	53776.8234	231.8983	0.8678	1.0948	0.4088
9	139.6237	47468.9443	217.8737	0.8835	0.9257	0.6486
Mean	142.2863	50290.7594	224.0257	0.8793	0.9007	0.5006
SD	5.1285	4572.1001	10.1618	0.0094	0.0857	0.0935

After tuning

	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	149.4252	55307.6031	235.1757	0.8720	1.0926	0.7191
1	166.1948	67242.4520	259.3115	0.8441	0.8796	0.6116
2	154.9602	56897.3477	238.5316	0.8636	0.8367	0.4720
3	145.6913	52194.4187	228.4610	0.8798	0.7979	0.4299
4	145.0494	49749.5333	223.0460	0.8776	0.9413	0.4976
5	152.2731	56243.7258	237.1576	0.8719	1.0030	0.4999
6	154.2157	57741.0009	240.2936	0.8616	0.9711	0.5118
7	141.1185	44182.8388	210.1971	0.8823	0.8177	0.6762
8	148.9689	58887.2391	242.6669	0.8552	1.1217	0.4403
9	148.5531	54608.7227	233.6851	0.8660	1.0150	0.8164
Mean	150.6450	55305.4882	234.8526	0.8674	0.9477	0.5675
SD	6.5658	5739.4449	12.2367	0.0112	0.1076	0.1247

CatBoost





- High density of points close to origin & low density of points away from the origin
- Symmetric about the origin

CatBoost

- Predicting on test sample

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	CatBoost Regressor	136.0082	46029.1762	214.5441	0.8887	0.9510	0.4791

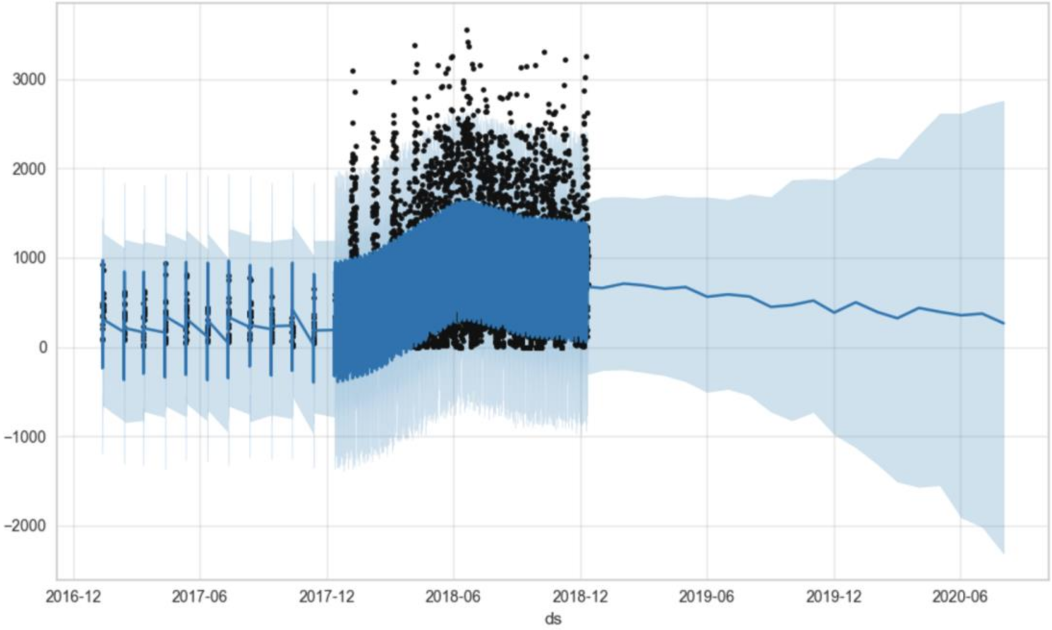
- Finalising the model

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	CatBoost Regressor	108.4352	27267.6359	165.1291	0.9341	0.8693	0.3765

Time Series ML - Prophet vs Long Short Term Memory Network(LSTM)

- WHY LSTM?
  - Regression problem & time series as we want to predict the trend of the rental bike after 2018
  - Subset of Recurrent Neural Network (RNN) suitable for time series & regression problem
  - Capable of learning long-term dependencies (remembering information for long period of time)
- WHY Prophet?
  - It works best with time series that have strong seasonal effects
  - Prophet is robust to outliers, missing data, and dramatic changes in time series.
  - Takes into account the data seasonality

Time Series - Prophet

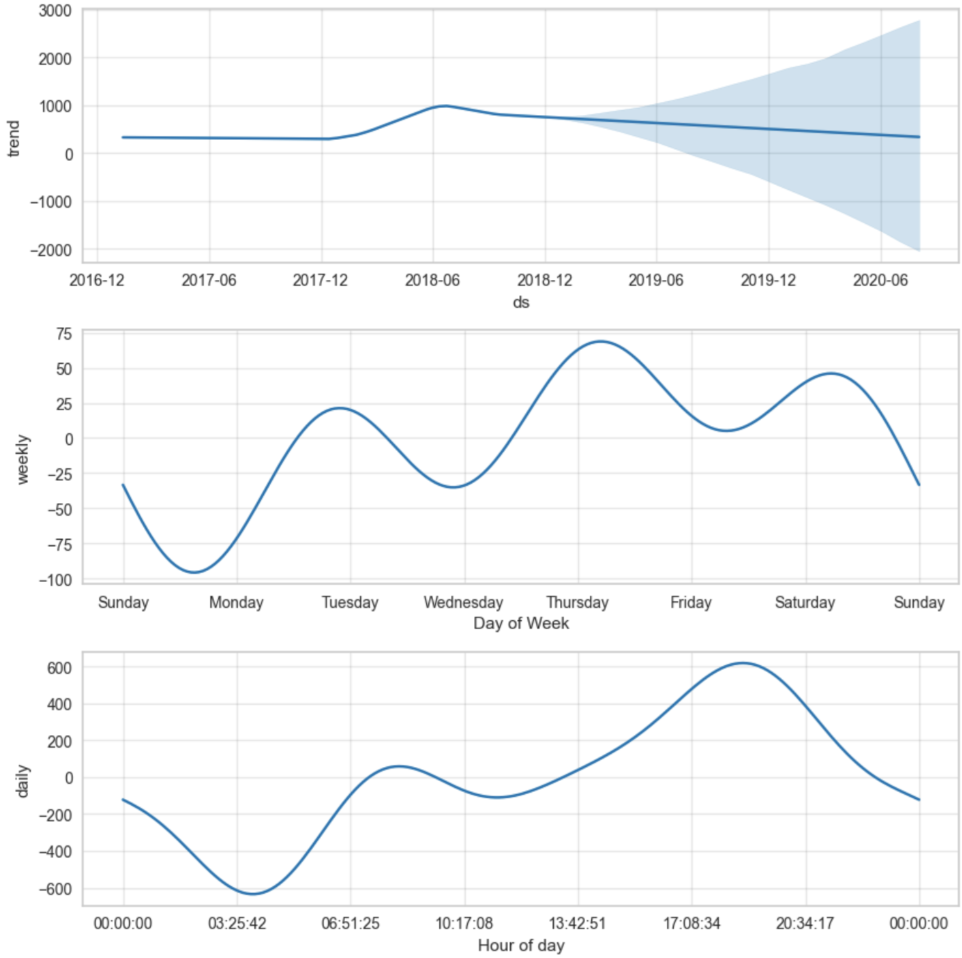


- In the graph the black dotted points represent the historical training data points.
- The blue line represents the forecasts generated for both history and future.
- Light blue region represents the uncertainty bands.

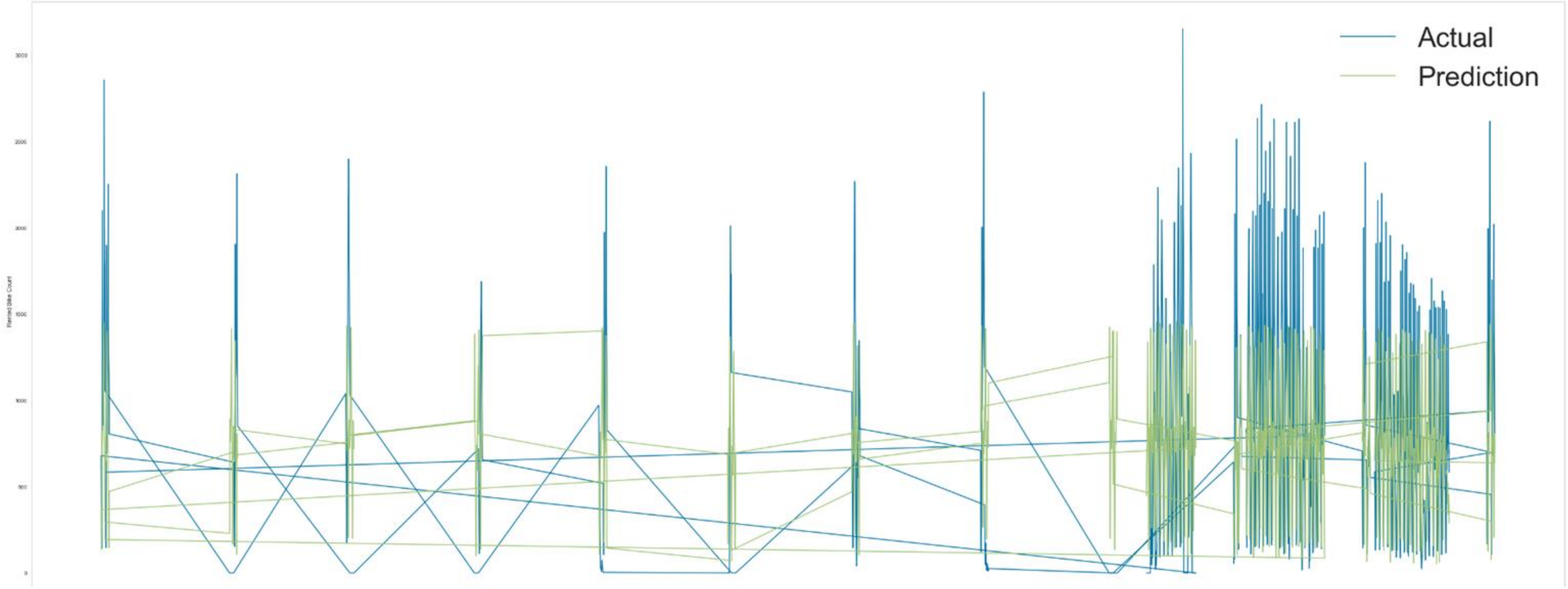
Time Series - Prophet

- Trend, weekly and daily seasonality of the time series

```
model.plot_components(forecast);
```



Prophet - Actual vs Prediction Plot



Index problem due to timestamp

Prophet - Final Prediction

```
prophet_ap_df.index=prophet_ap_df.index.strftime('%d/%m/%Y, %r')
prophet_ap_df
```

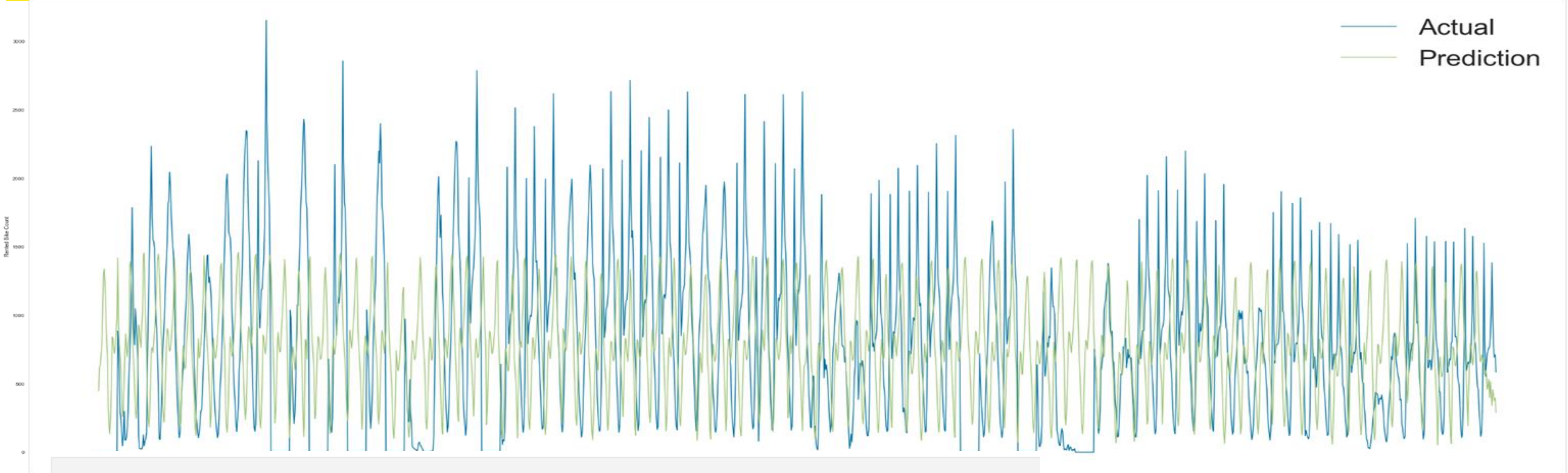
✓ 0.1s

	Actual	Prediction
Timestamp		
19/09/2018, 12:00:00 AM	0	447.163685
19/09/2018, 01:00:00 AM	0	620.296946
19/09/2018, 02:00:00 AM	0	633.377791
19/09/2018, 03:00:00 AM	0	700.261267
19/09/2018, 04:00:00 AM	0	786.216242
...	...	...
30/11/2018, 07:00:00 PM	1003	455.928825
30/11/2018, 08:00:00 PM	764	413.091362
30/11/2018, 09:00:00 PM	694	376.156695
30/11/2018, 10:00:00 PM	712	396.167130
30/11/2018, 11:00:00 PM	584	288.886597

1752 rows × 2 columns

Change the timestamp from ‘Datetime’ index to string

Prophet - Actual vs Prediction plot

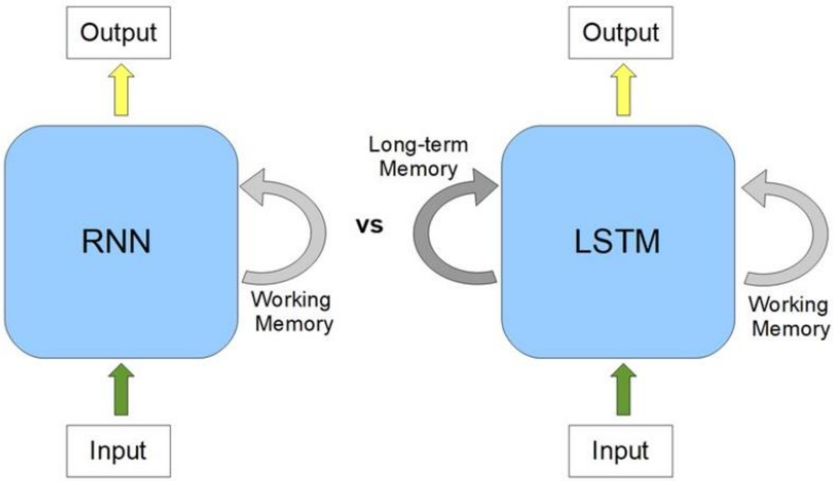


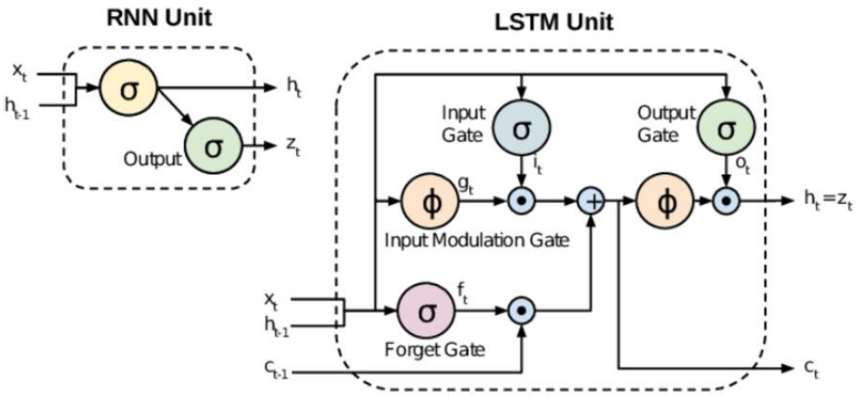

```
mse_Prophet = mean_squared_error(prophet_ap_df['Prediction'],prophet_ap_df['Actual'])
mae_Prophet = mean_absolute_error(prophet_ap_df['Prediction'],prophet_ap_df['Actual'])
mape_Prophet = mean_absolute_percentage_error(prophet_ap_df['Prediction'],prophet_ap_df['Actual'])
r2_Prophet = r2_score(prophet_ap_df['Prediction'],prophet_ap_df['Actual'])
print("The Mean Square Error for Prophet Model is ",round(mse_Prophet,4))
print("The Mean Absolute Error for Prophet Model is ",round(mae_Prophet,4))
print("The Mean Absolute Percentage Error for Prophet Model is ",round(mape_Prophet,4))
print("The R2 score for Prophet model is",round(r2_Prophet,4))
```

✓ 0.1s

```
The Mean Square Error for Prophet Model is  511300.6998
The Mean Absolute Error for Prophet Model is  580.1104
The Mean Absolute Percentage Error for Prophet Model is  1.1013
The R2 score for Prophet model is -3.3763
```

LSTM VS RNN





Three types of gates within LSTM unit:

- **Forget Gate**: conditionally decides what information to throw away from the block
- **Input Gate**: conditionally decides which values from the input to update the memory state
- **Output Gate**: conditionally decides what to output based on input and the memory of the block

LSTM DATA PREPARATION

Use Univariate Time Series Forecasting on LSTM to get the output of rental bike count

```
#Train Test split before LSTM (80% Train 20% Test)
train_LSTM = encoded_df.iloc[:7008,-3].values
test_LSTM = encoded_df.iloc[7008:,-3].values
```

✓ 0.5s

```
train_LSTM=np.reshape(train_LSTM,(7008,1))
test_LSTM=np.reshape(test_LSTM,(1752,1))
```

✓ 0.5s

```
#Standardize data between 0 and 1
sc = MinMaxScaler()
train_LSTM_scaled = sc.fit_transform(train_LSTM)
test_LSTM_scaled = sc.transform (test_LSTM)
```

✓ 0.1s

```
X_train=[]
Y_train=[]
X_test=[]
Y_test=[]

#50 steps for one input(1 value) for train set
for i in range(50, len(train_LSTM_scaled)):
    X_train.append(train_LSTM_scaled[i-50:i,0])
    Y_train.append(train_LSTM_scaled[i,0])
X_train, Y_train = np.array(X_train), np.array(Y_train)

#using final 50 steps from train set to predict output on testing
test_LSTM_scaled_50=np.vstack((train_LSTM_scaled[-50:],test_LSTM_scaled))

#50 steps for one input(1 value) for test set
for i in range(50, len(test_LSTM_scaled_50)):
    X_test.append(test_LSTM_scaled_50[i-50:i,0])
    Y_test.append(test_LSTM_scaled_50[i,0])
X_test, Y_test = np.array(X_test), np.array(Y_test)
```

✓ 0.7s

- Train test split (80% Train 20% Test)
  Total data = 8760 rows
- Use MinMaxScaler to reduce outliers for the models to be trained & tested
- 50 data as 1 input for 1 output in Train & Test sets before LSTM Model.

LSTM - Set Layer Network

```
LSTM_regression = Sequential() #initialising model
```
✓ 0.2s

```
# Add first layer of neural network
# units: number of neurons in hidden layer - put 100 neurons into it
# activation: activation function to be used, Sigmoid or tanh
# input shape: data shape to be provide for LSTM RNN
# fix dropout rate at 20%
LSTM_regression.add(LSTM(units=100, return_sequences=True, input_shape=(X_train.shape[1],1)))
LSTM_regression.add(Dropout(0.2))
```

```
# Add second layer of neural network similar to first layer
LSTM_regression.add(LSTM(units=100, return_sequences=True))
LSTM_regression.add(Dropout(0.2))
```
✓ 0.4s

```
# Add Final Layer before output similar to first layer
LSTM_regression.add(LSTM(units=100))
LSTM_regression.add(Dropout(0.2))
```
✓ 0.3s

```
# Output layer
LSTM_regression.add(Dense(units=1))
```
✓ 0.1s

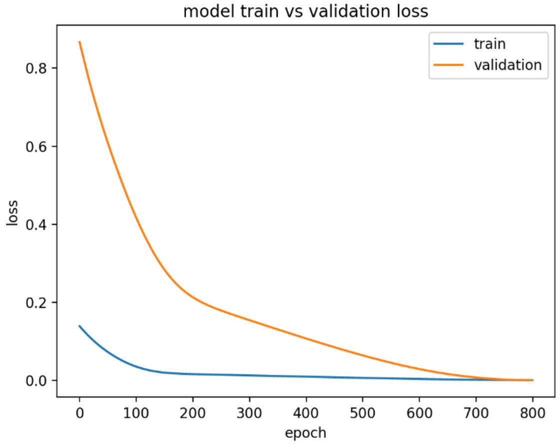
```
LSTM_regression.summary()
```
✓ 0.1s

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 50, 100)           40800

 dropout (Dropout)           (None, 50, 100)           0

 lstm_1 (LSTM)               (None, 50, 100)           80400

 dropout_1 (Dropout)         (None, 50, 100)           0

 lstm_2 (LSTM)               (None, 100)               80400

 dropout_2 (Dropout)         (None, 100)               0

 dense (Dense)               (None, 1)                 101

=================================================================
Total params: 201,701
Trainable params: 201,701
Non-trainable params: 0
_________________________________________________________________
```

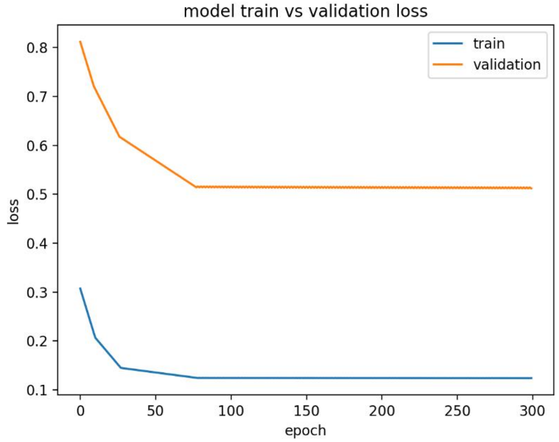
- Total 3 Layer Network before final output
  - First Layer Network - 40800 parameters
  - Second Layer Network - 80400 parameters
  - Third Layer Network - 80400 parameters
  - Final Output Layer - 101 parameters

- Total trainable parameters: 201701

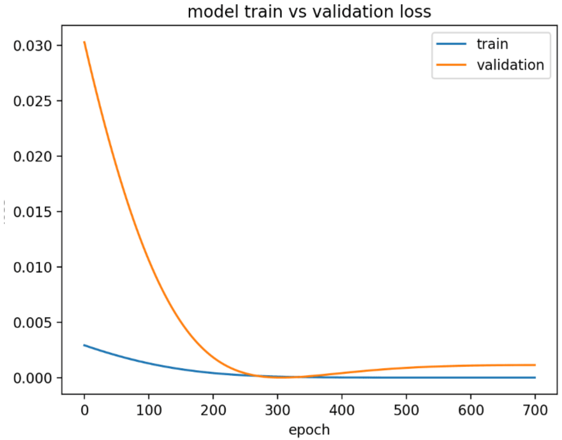
LSTM - Diagnostic Plot Example



Diagnostic Line Plot Showing a Good Fit for a Model



Diagnostic Line Plot Showing an Underfit Model via Status



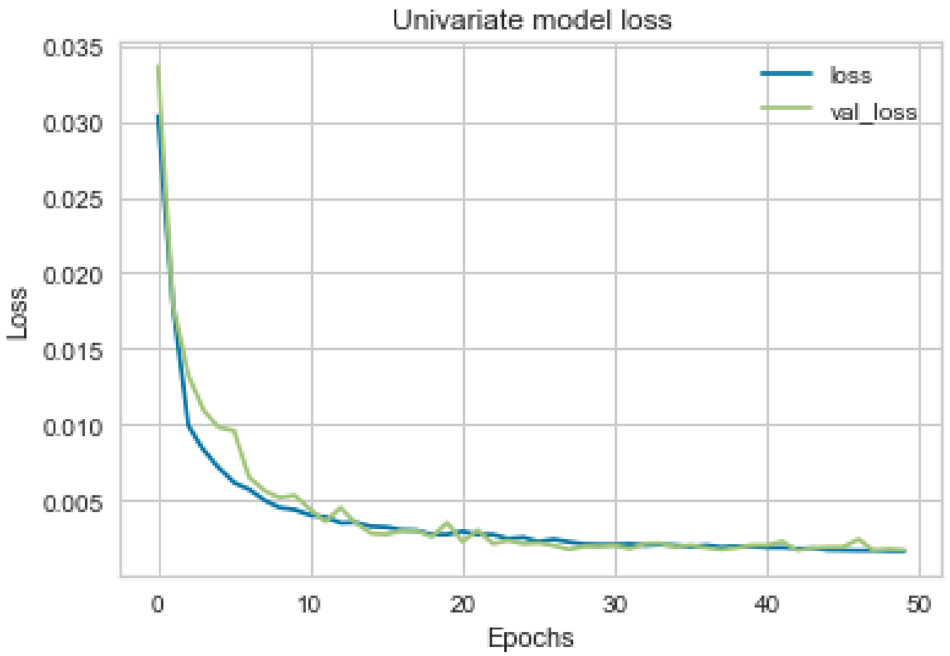
Diagnostic Line Plot Showing an Overfit Model

Good Fit

Over Fit

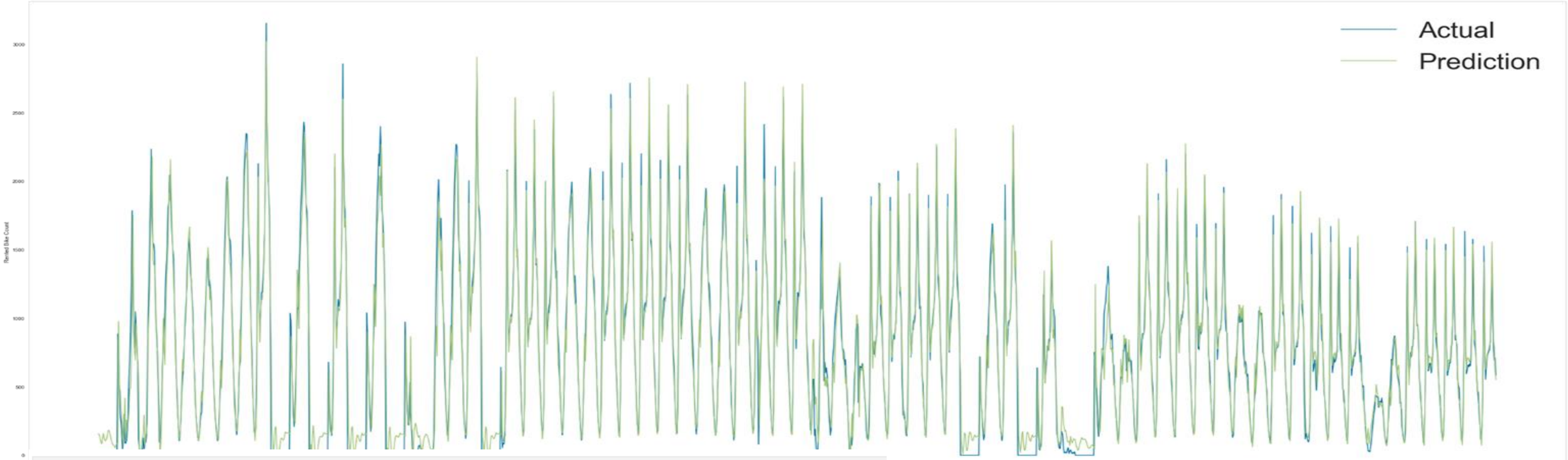
Under Fit

LSTM - Diagnostic Plot Example



- Good fit for the data in LSTM Model

LSTM - Final Prediction




```python
mse_LSTM = mean_squared_error(prediction,Y_test)
mae_LSTM = mean_absolute_error(prediction,Y_test)
mape_LSTM = mean_absolute_percentage_error(prediction,Y_test)
r2_LSTM = r2_score(prediction, Y_test)
print("The Mean Square Error for LSTM Model is ",round(mse_LSTM,4))
print("The Mean Absolute Error for LSTM Model is ",round(mae_LSTM,4))
print("The Mean Absolute Percentage Error for LSTM Model is ",round(mape_LSTM,4))
print("The R2 score for LSTM model is",round(r2_LSTM,4))
```

✓ 0.7s

```
The Mean Square Error for LSTM Model is  0.0017
The Mean Absolute Error for LSTM Model is  0.0272
The Mean Absolute Percentage Error for LSTM Model is  0.2956
The R2 score for LSTM model is 0.9375
```

Conclusion

Use Mean Absolute Percentage Error (MAPE) as a tool to check which model is suitable for Machine Learning Model.

1. Catboost - 0.37
2. Light GBM - 0.41
3. Prophet - 1.10
4. LSTM - 0.29

Final Model Use For ML : LSTM - MAPE 0.29

We can recommend Company A to invest in bike rental as annual bike rental count is 6.1 million. ML model also predicted higher usage of rental bike after year 2018.

Thank you!
Q&A

GitHub link:

[evansim85/Capstone-Project: Seoul Bike Sharing Demand Prediction (github.com)]

[https://github.com/sinyihehe/DS106]