Machine Learning for Hourly Bike Sharing Prediction

Goal: To predict the hourly number of bicycles shared based on the given features. Please see Jupyter Notebook for the code for each of the analysis.

We will break our report into following sections.

Sections:

1. Data
2. Definition of fields
3. Exploratory Data Analysis
    a. Preprocessing steps
    b. Missing value analysis
    c. Correlation
    d. Y variable analysis
    e. Bivariate analysis and Kruskal Wallis test statistic
    f. Insights and Decisions from EDA
4. Modelling, Results and Analysis of Errors
5. Bonus Material
6. Production Code (main.py) specifics

Data: Link

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	0	6	0	1	0.24	0.2879	0.81	0.0	3	13	16
1	2	2011-01-01	1	0	1	1	0	6	0	1	0.22	0.2727	0.80	0.0	8	32	40
2	3	2011-01-01	1	0	1	2	0	6	0	1	0.22	0.2727	0.80	0.0	5	27	32
3	4	2011-01-01	1	0	1	3	0	6	0	1	0.24	0.2879	0.75	0.0	3	10	13
4	5	2011-01-01	1	0	1	4	0	6	0	1	0.24	0.2879	0.75	0.0	0	1	1

Definition of fields:

1. Instant: Record Index
2. Date day: Date-time timestamp
3. Season: (1: Spring, 2: Summer, 3: Fall, 4: Winter)
4. Year: (0: 2011, 1:2012)
5. Month: (1 to 12)
6. Hour (0 to 23)
7. Holiday: (0 or 1)
8. Weekday: (0 to 6)
9. Working day: (0 or 1)
10. Weather Situation (1: Clear, Partly Cloudy
    2: Mist + Cloudy

3: Light Snow, light rain
4: Heavy rain, ice pallets)

11. Temperature (Normalized, Celsius)
12. Feeling Temperature (Normalized, Celsius)
13. Humidity (Normalized)
14. Wind Speed (Normalized)
15. Casual Users who rented bikes (Count)
16. Registered Users who rented bikes (Count)
17. Total Count of users (Casual + Registered) who rented bike

Exploratory Data Analysis:

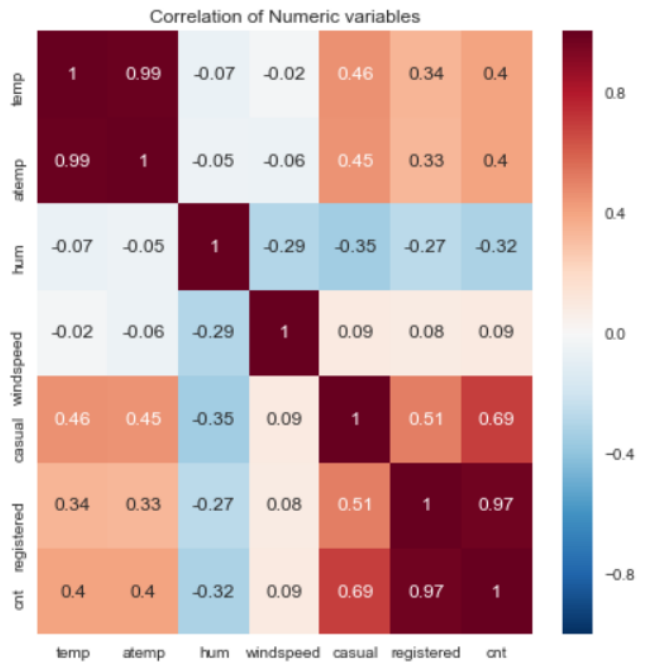
The goal of this section is to analyze the data, see the relationship between the features, see the distributions, etc. to make relevant decisions.

The type of decisions we can make from EDA is:
1. What features we want to keep / drop
2. What rows we want to keep / drop for training machine learning model
3. How do we handle missing values
4. Deciding what algorithm may work according to the data

Preprocessing steps
We indexed the data on 'instant' column and converted season, year, month, hour, holiday, weekday, working day and weather situation into categorical attributes.

Missing Value Analysis
We have no missing values in the data.

Correlation between Numeric variables



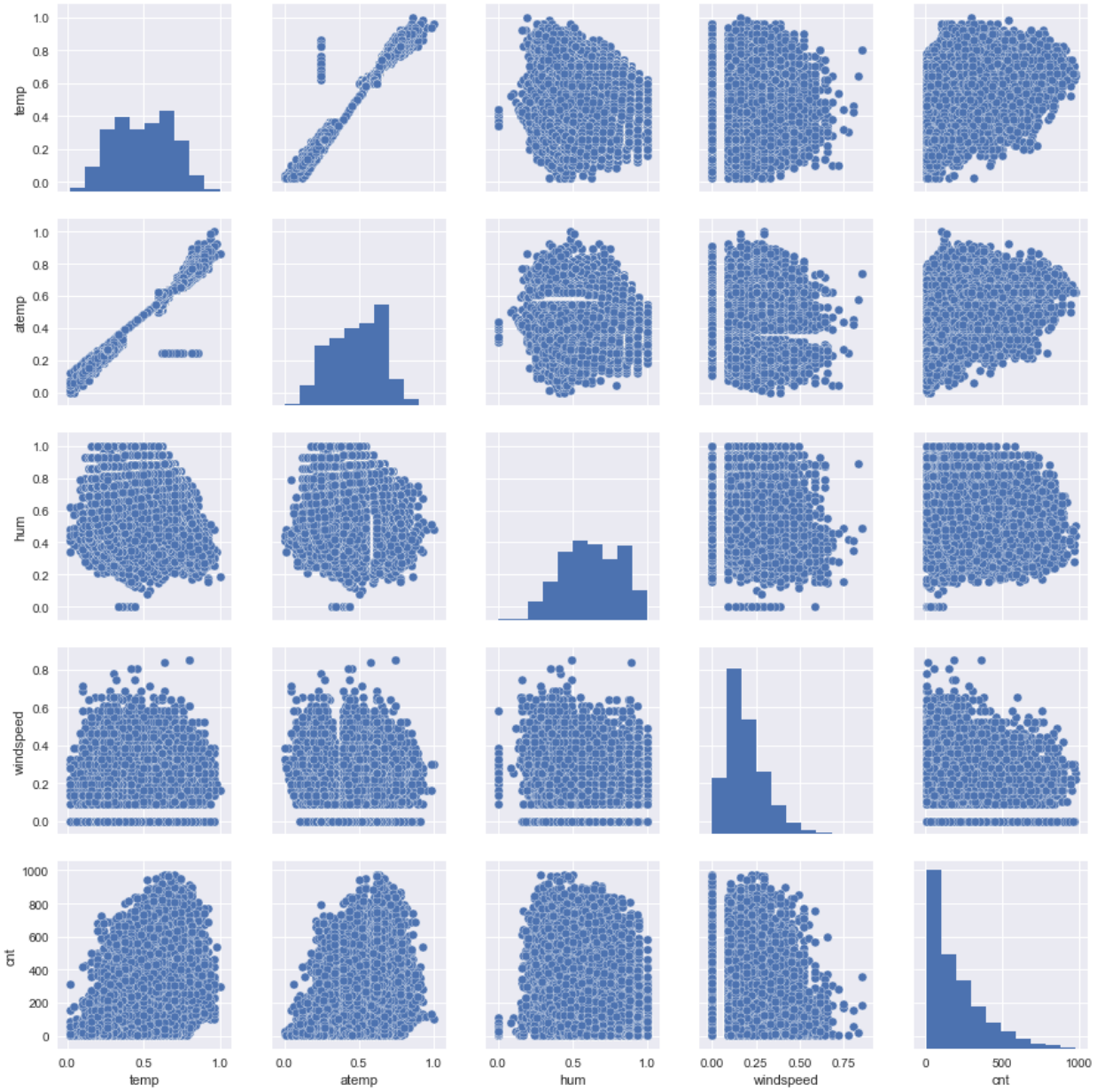
We see that Casual Users count and Registered Users count are correlated to the Hourly count, because Total users = casual users + registered users.
But we **don't want to use** these 2 features.

The other features are not well correlated with the Y variable.
Since we have less variables, we can see a pair plot.

Basic Numeric Statistics

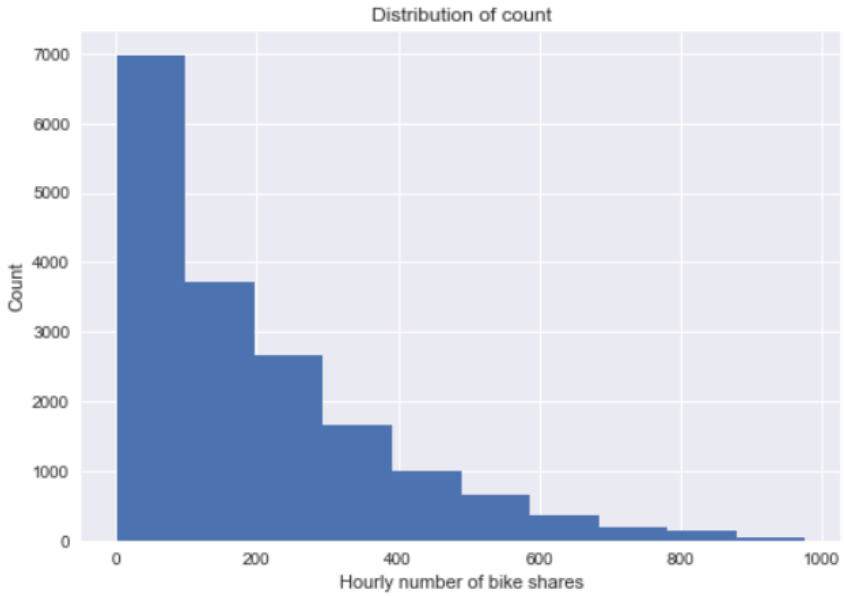
	temp	atemp	hum	windspeed	casual	registered	cnt
count	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000
mean	0.496987	0.475775	0.627229	0.190098	35.676218	153.786869	189.463088
std	0.192556	0.171850	0.192930	0.122340	49.305030	151.357286	181.387599
min	0.020000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	0.340000	0.333300	0.480000	0.104500	4.000000	34.000000	40.000000
50%	0.500000	0.484800	0.630000	0.194000	17.000000	115.000000	142.000000
75%	0.660000	0.621200	0.780000	0.253700	48.000000	220.000000	281.000000
max	1.000000	1.000000	1.000000	0.850700	367.000000	886.000000	977.000000

We see that features are normalized in some way (as mentioned in the data description).



From the above picture, we can see that the variables are **not normally distributed**.

Y variable Analysis



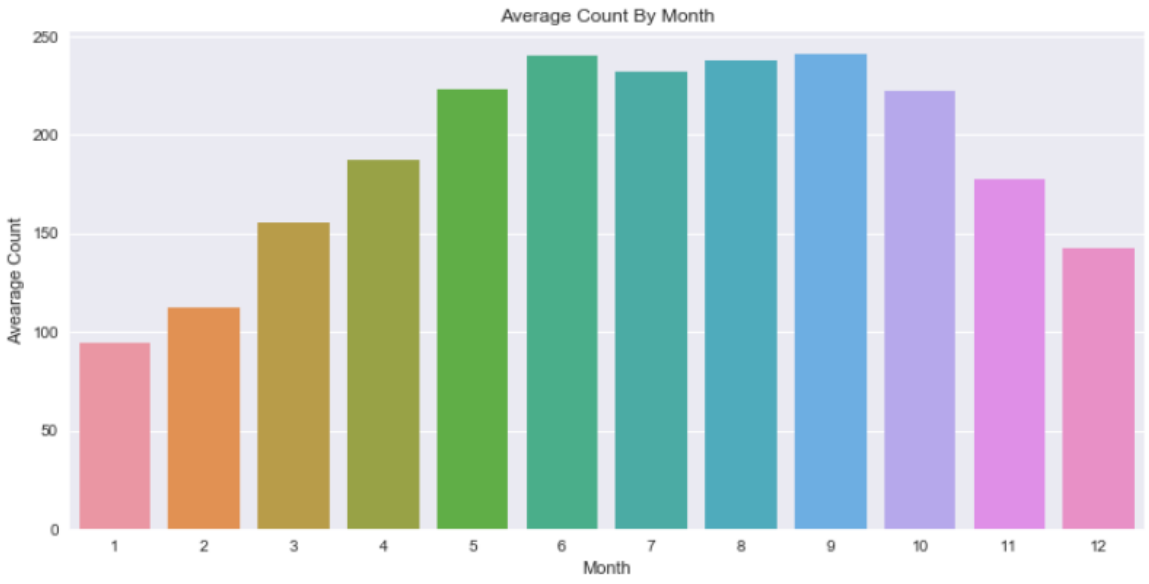
We see that the distribution of Y variable is right skewed.

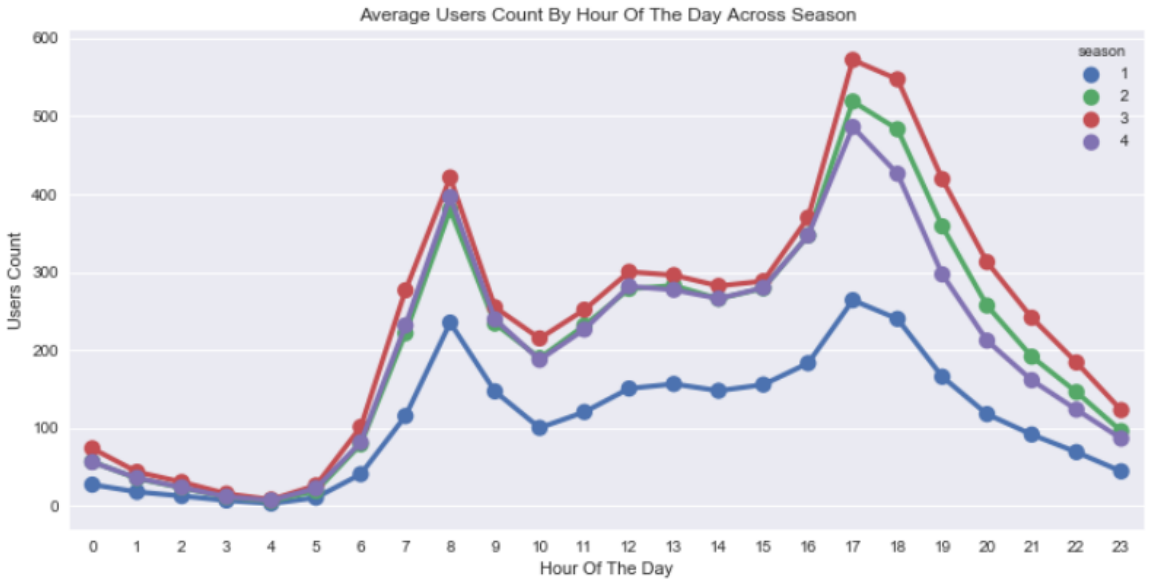
Value Counts

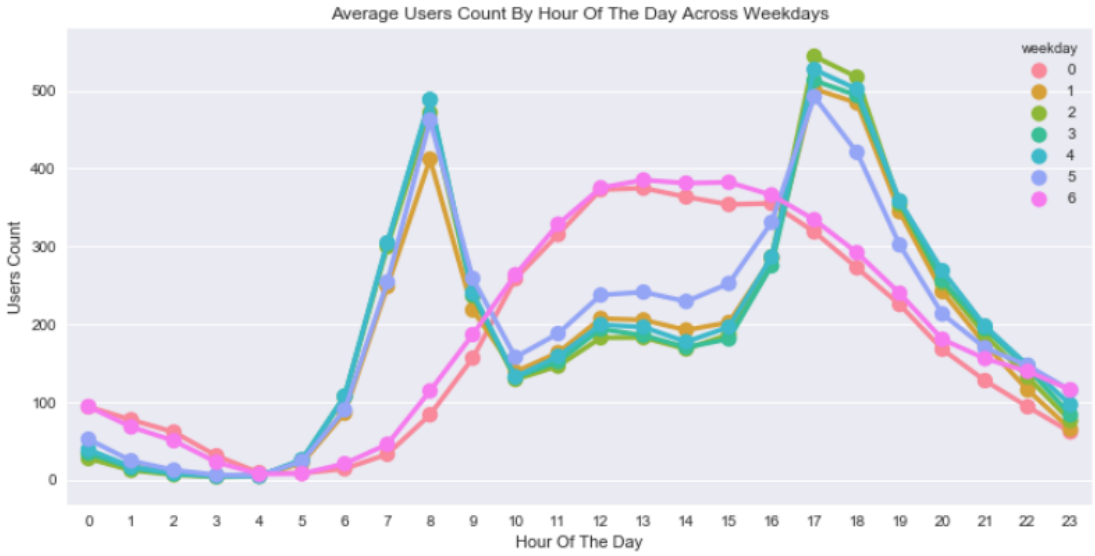
We observe that the Weather Situation column is highly unbalanced.
Type 1: 11413 observations
Type 2: 4544 observations
Type 3: 1419 observations
Type 4: 3 observations

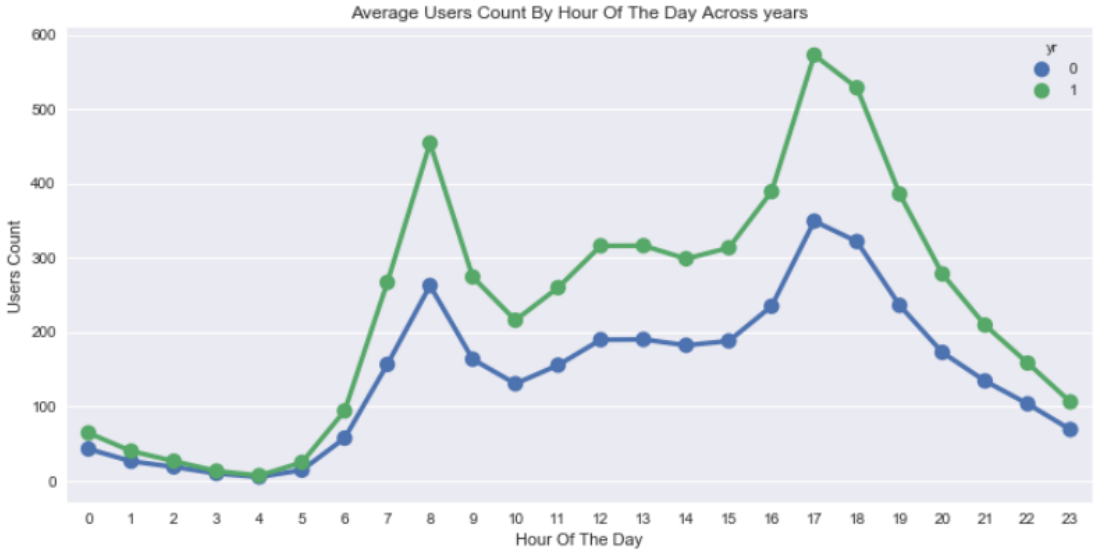
Bivariate Analysis

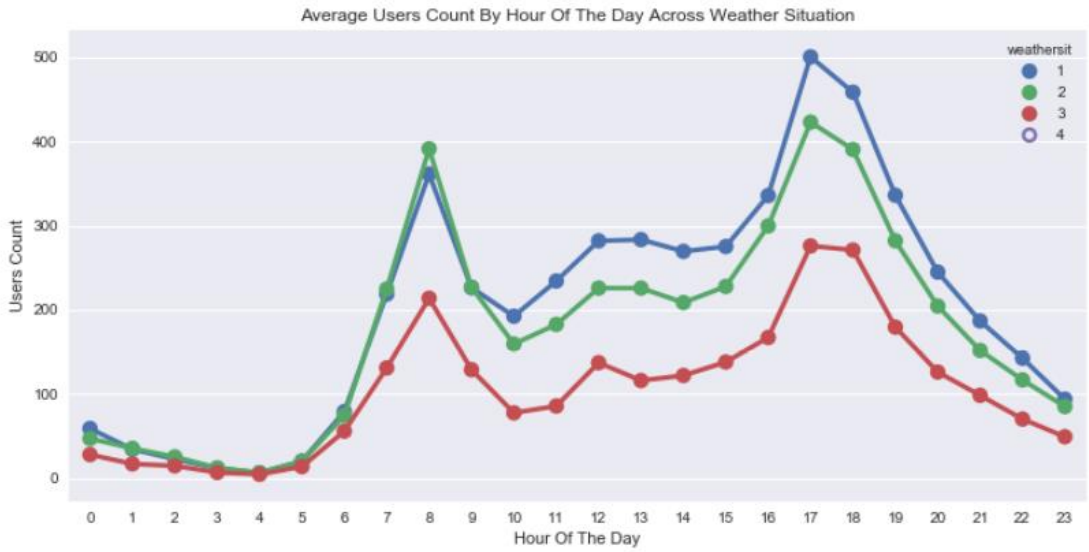
Comparing various categorical features along the hour with the Hourly count (see the code in Notebook):

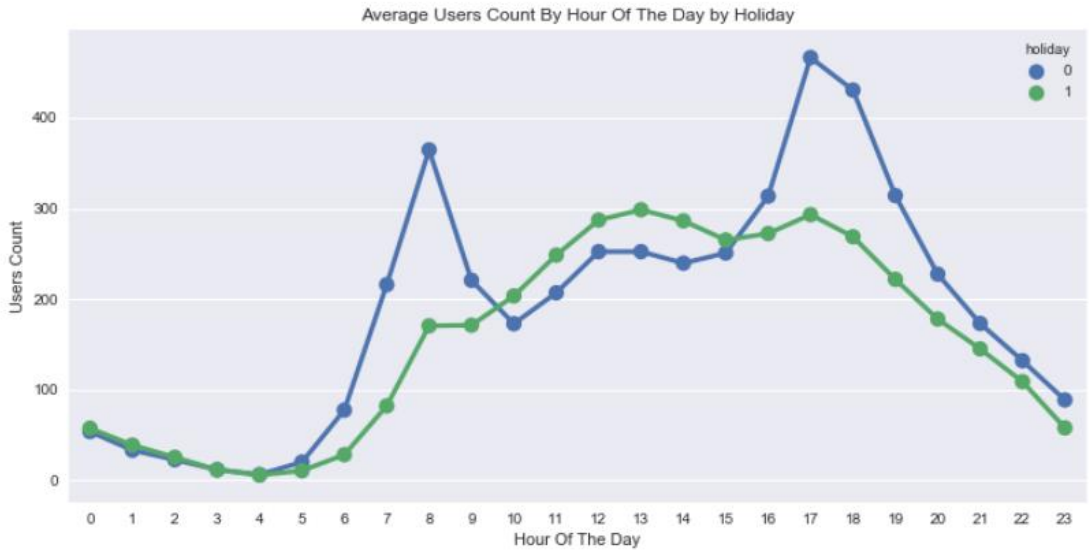

Average Count By Month
Avearage Count
Month
0
50
100
150
200
250
1
2
3
4
5
6
7
8
9
10
11
12


Average Users Count By Hour Of The Day Across Season
Users Count
Hour Of The Day
season
1
2
3
4
0
100
200
300
400
500
600
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23


Average Users Count By Hour Of The Day Across Weekdays
weekday
0
1
2
3
4
5
6
Users Count
0
100
200
300
400
500
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour Of The Day


Average Users Count By Hour Of The Day Across years
yr
0
1
Users Count
0
100
200
300
400
500
600
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour Of The Day





From the graphs, the **hour** of the day and the **month** seem to be the most important categorical features.
We can also verify this by Kruskal Wallis test (because Y variable is not normal, we cannot use ANOVA):

<u>P value from Kruskal Wallis test</u>
Season = 9.47 e-258
Year = 9.7 e-165
Month = 4.96 e-270
Hour = 0.0
Holiday = 9.5 e-5
Weekday = 0.0004

Working Day = 0.0055
Weather Situation = 9.7 e-86

The hour and the Month features have the lowest Kruskal Wallis P-value, meaning they are statistically most significant variables to predict Y variable.

Insights and Decisions

Following are the insights we have gained so far:

1. No numeric feature is significantly correlated.
2. Numeric features are not normally distributed.
3. Even though numeric features are not normally distributed, they are already standardized.
4. Hour and month are the most significant categorical features.
5. Weather situation is imbalanced.
6. There are no missing values.

Based on the above insights, we can make following decisions.

1. Drop rows with Weather situation = 4.
2. Drop columns Casual, Registered and Date-time.
3. Since we don't see linear correlation with Y variable and the predictors, and the predictors are not normally distributed, parametric models with high bias (Linear Regression, Ridge, Lasso) would tentatively fail.
   The non-parametric models like decision tree with possible Ensemble technique (Bagging or Boosting), or parametric model which learns non-linear relationship (Neural Networks) may succeed.

Modelling and Results

Model selection and Reasons

1. As mentioned above in the Insights and Decisions section of EDA, the parametric models like Linear Regression, and its regularized versions would tentatively fail because they have strong assumptions.
2. The features are not normally distributed, and the relationship doesn't seem linear (as seen from correlation matrix heatmap).
3. Multicollinearity exists between two features of temperature.
4. Since parametric model with strong assumptions would not work, I will choose a **flexible model**.
5. I will choose **Gradient Boosting Trees** and present my analysis.
6. The gradient boosting is non-parametric method, and can produce flexible models. They do **Boosting ensemble** across decision trees.
7. It works very well for non normal data, in which the Linear regression would fail because of high bias. Gradient boosting may overfit if you add too many classifiers / trees.

Model Hyperparameters (method to select best hyperparameters)

The primary hyperparameters for Gradient Boost Regressor are **number of estimators**, **depth** of each estimator (decision tree) and **learning rate**.

I evaluated the choice of hyperparameters by **Grid-search-CV** module, which basically iterates over different hyperparameters, performs **Cross Validation**, and chooses the parameters which give the best result in the **hold out / validation** sets.

Some parameters like Maximum features to consider wasn't tuned, because we have less number of features.

The best hyperparameters:

Number of Estimators = 200
Max depth of Estimator = 10
Learning Rate = 0.1

Model Evaluation

We evaluate the model by Mean Absolute Deviation / Error, and R square (used from in built Scikit-learn). We don't use Adjusted R squared because the dimension would increase with increase in levels of categorical features in training data.

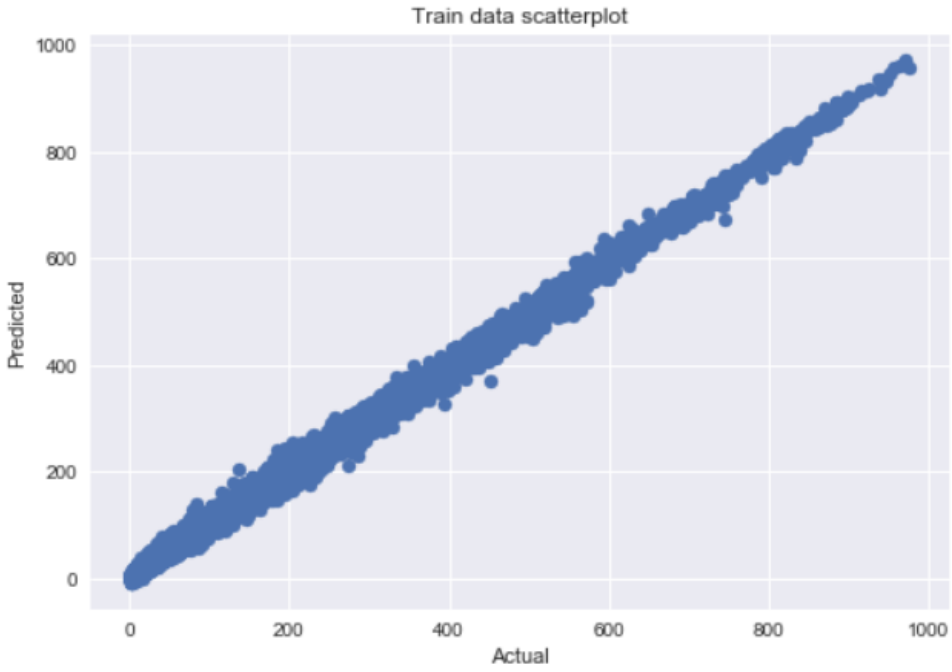
Model Results

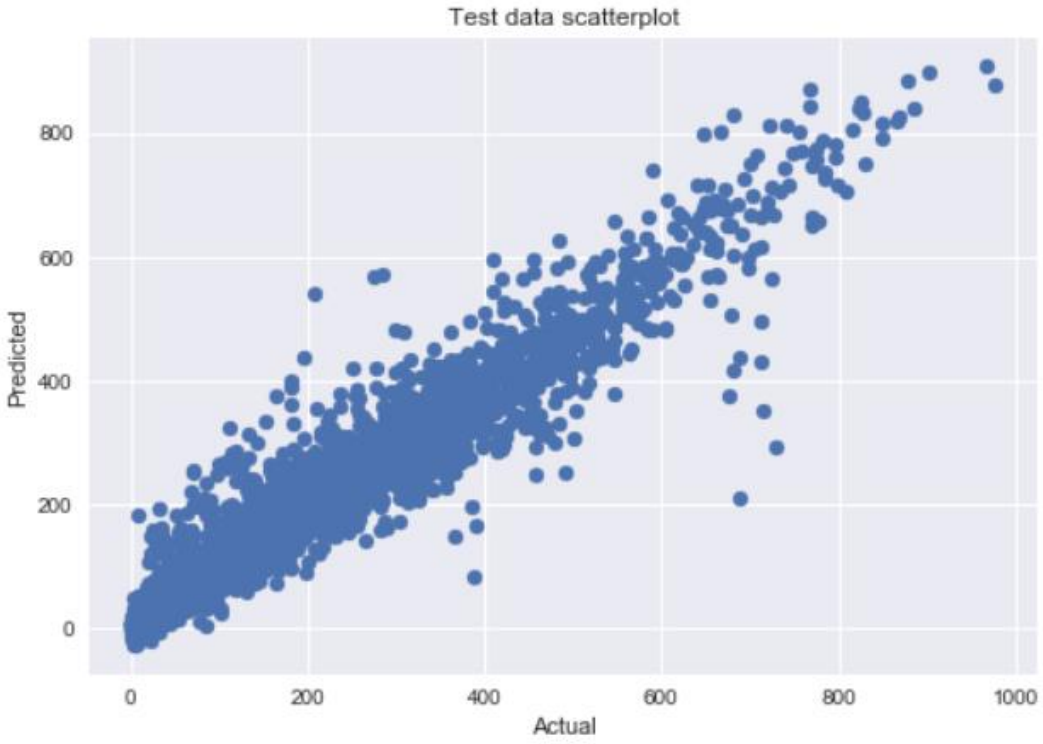
Model Name	**Train R Squared**	**Test R Squared**	**Train MAE**	**Test MAE**
Gradient Boost	0.99	0.92	6.66	28.99

Please note that the best model has been chosen from **Cross validation**.

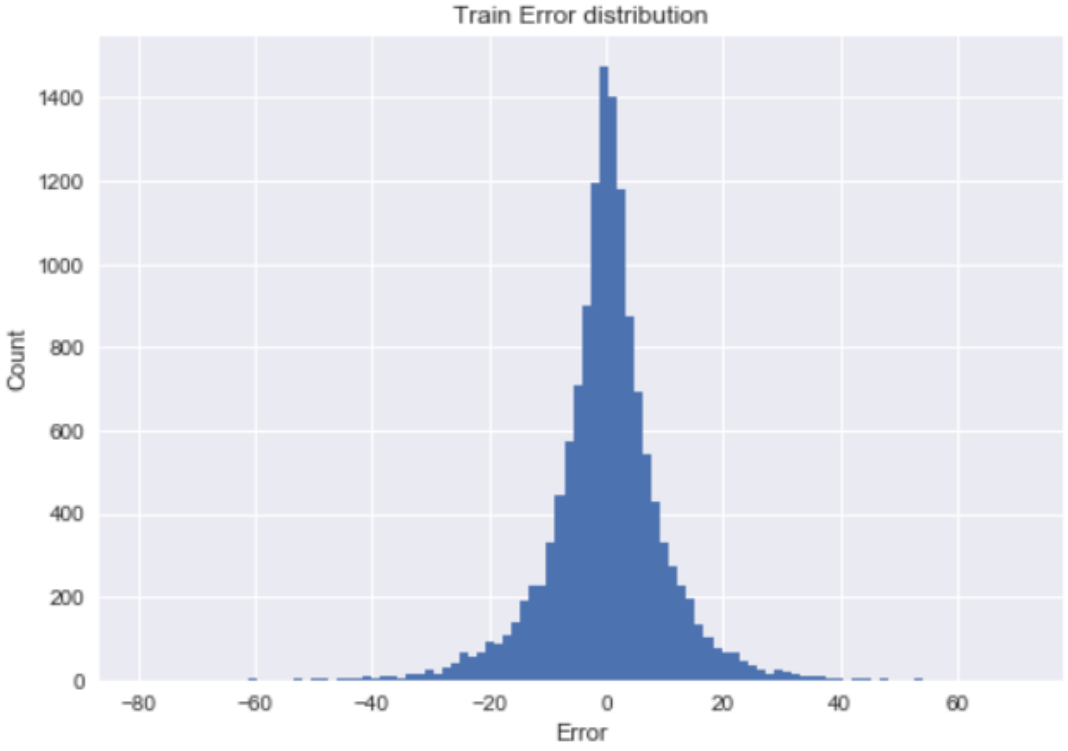
Plots of Results

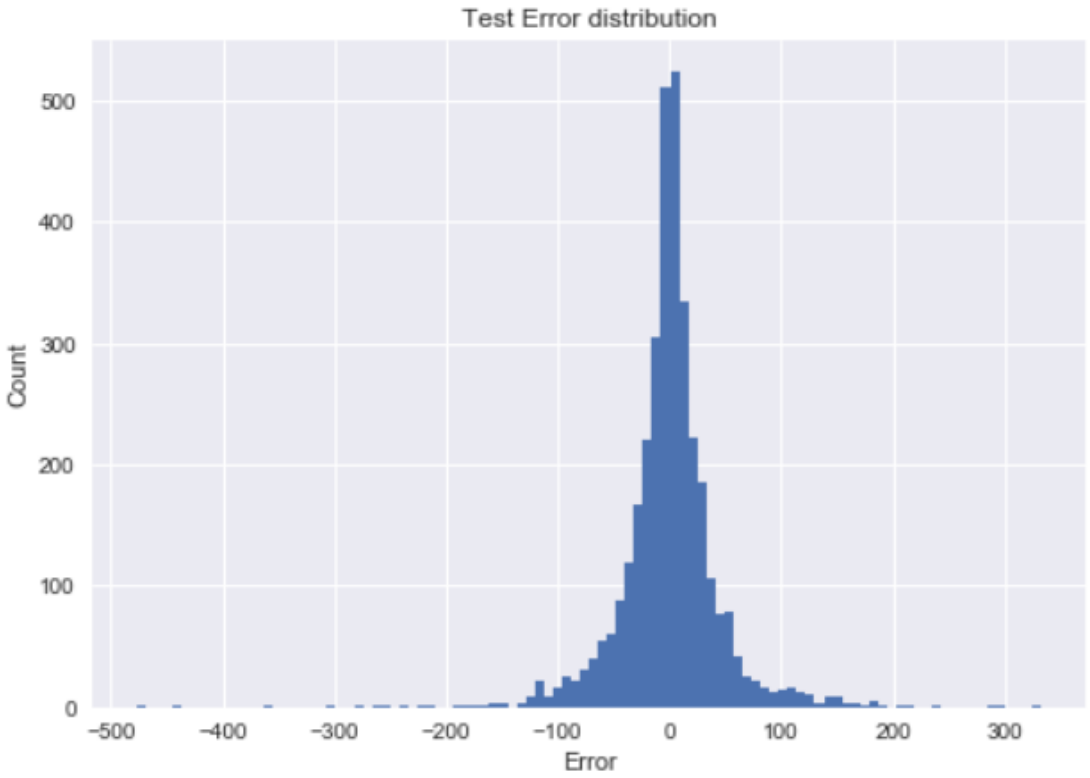
We have plots of Y predicted vs Y actual in train and test data, and plot of distribution of errors.





We also show the plots of errors.





We see that the errors are approximately normally distributed.

Bonus Material

We use GridSearchCV over other models and evaluate the results and the best parameters from each model.

	Best tuned model	Train Accuracy	Test Accuracy	Train MAE	Test MAE	Best hyper parameters
0	Linear Regression	0.686322	0.684805	75.811906	73.795473	[]
1	Lasso Regression	0.683785	0.682160	75.630852	73.830186	[["regressor__alpha", 0.3]]
2	Gradient Boost Regressor	0.989862	0.926448	12.571981	30.266520	[["regressor__learning_rate", 0.1], ["regressor__max_depth", 10], ["regressor__n_estimators", 100]]
3	Random Forest Regressor	0.810038	0.775654	58.392682	62.583509	[["regressor__max_depth", 10], ["regressor__n_estimators", 100]]
4	Neural Networks	0.921758	0.915621	32.993188	33.459383	[["regressor__hidden_layer_sizes", [10, 10, 10]]]

As expected, the linear Regression and Lasso Regression perform poorly.
Random Forest does slightly better.
But the best accuracies are achieved by **Gradient Boosting** and **Neural Networks**, which are flexible models in nature.

About Production Code (main.py):

We can pass arguments to main.py as:
Training: python main.py trainFileName –train
Testing: python main.py testFileName

During training, cross validation results on Train data is shown. Model is saved in pickle file (model.pkl), and 2 plots are saved (Scatterplot of Y predicted vs Y actual, histogram of errors).

During testing phase, the predictions of each row is saved in TestDataPredictions.csv, and if we have the information of Y variable in test data then accuracy is calculated and respective plots are saved.