BIKE SHARING DATASET ANALYSIS

IST 652 SCRIPTING FOR DATA ANALYSIS

Abstract

Perform EDA, Data pre-processing and visualizations to analyze the dataset and create models, supervised and unsupervised models for prediction and classification.

Sushil Sunilkumar Deshmukh

578130808

sdeshmuk@syr.edu

TABLE OF CONTENTS

BIKE SHARING DEMAND

About the data

Bike sharing systems are a means of renting bicycles where the process of obtaining membership, rental, and bike return is automated via a network of kiosk locations throughout a city. Using these systems, people are able rent a bike from a one location and return it to a different place on an as-needed basis. Currently, there are over 500 bike-sharing programs around the world.

The data generated by these systems makes them attractive for researchers because the duration of travel, departure location, arrival location, and time elapsed is explicitly recorded. Bike sharing systems therefore function as a sensor network, which can be used for studying mobility in a city. In this competition, participants are asked to combine historical usage patterns with weather data in order to forecast bike rental demand in the Capital Bikeshare program in Washington, D.C.

The dataset can be found at this URL : https://www.kaggle.com/c/bike-sharing-demand

There are two files that are:

1. Train.csv – This file contains all the complete data needed to train a model to make predictions.
2. Test.csv – This file contains the test data for the model to work with and increase the accuracy.

Data Import, Exploration and Pre-Processing

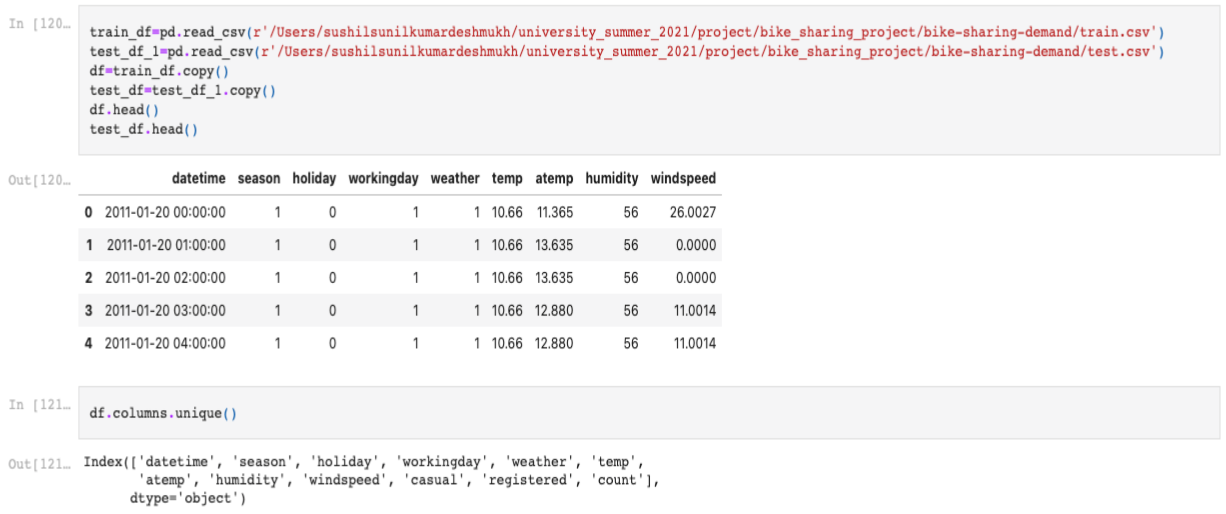
The train file and the test file are loaded into respective pandas dataframe for exploration. Here are some basic information about the data files.

1. There are total of 17377 rows in combined two files.
2. There are total of 12 columns or attributes that are variables in this dataset.

Different attributes for the dataset and their explanation of the categorical values are as follows:

- datetime - hourly date + timestamp
- season - 1 = spring, 2 = summer, 3 = fall, 4 = winter
- holiday - whether the day is considered a holiday
- workingday - whether the day is neither a weekend nor holiday
- weather - 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
- temp - temperature in Celsius
- atemp - "feels like" temperature in Celsius
- humidity - relative humidity
- windspeed - wind speed
- casual - number of non-registered user rentals initiated
- registered - number of registered user rentals initiated
- count - number of total rentals

Performing Exploratory Data analysis, We used the pandas library and plotted some graphs.



```
In [120…
train_df=pd.read_csv(r'/Users/sushilsunilkumardeshmukh/university_summer_2021/project/bike_sharing_project/bike-sharing-demand/train.csv')
test_df_1=pd.read_csv(r'/Users/sushilsunilkumardeshmukh/university_summer_2021/project/bike_sharing_project/bike-sharing-demand/test.csv')
df=train_df.copy()
test_df=test_df_1.copy()
df.head()
test_df.head()
```

Out[120…

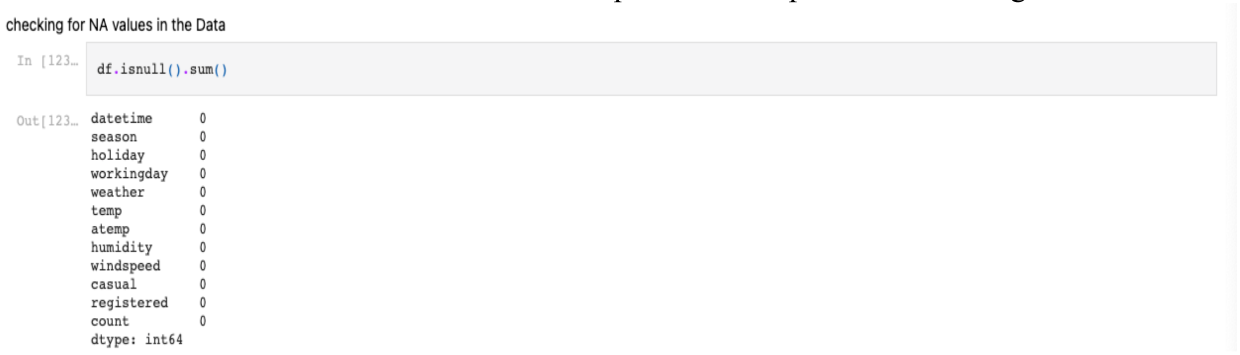
	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed
0	2011-01-20 00:00:00	1	0	1	1	10.66	11.365	56	26.0027
1	2011-01-20 01:00:00	1	0	1	1	10.66	13.635	56	0.0000
2	2011-01-20 02:00:00	1	0	1	1	10.66	13.635	56	0.0000
3	2011-01-20 03:00:00	1	0	1	1	10.66	12.880	56	11.0014
4	2011-01-20 04:00:00	1	0	1	1	10.66	12.880	56	11.0014

```
In [121…
df.columns.unique()

Out[121… Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count'],
      dtype='object')
```

Fig 1.

NA Values

The first step of data pre-processing is checking of there are any NA values and then eliminating them from the data since this can be an issue and hamper the model that we are developing.
The code used to check the NA values in the script and the output is as following:



checking for NA values in the Data

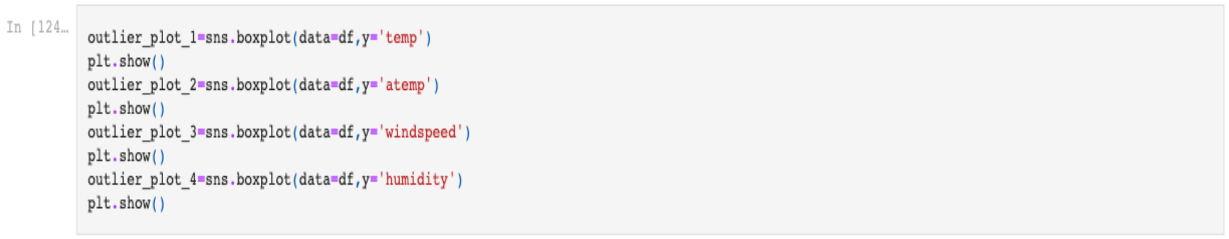
```
In [123…
df.isnull().sum()

Out[123… datetime      0
season        0
holiday       0
workingday    0
weather       0
temp          0
atemp         0
humidity      0
windspeed     0
casual        0
registered    0
count         0
dtype: int64
```

Fig 2.

We can see that NO NULL values can be found in this dataset. This means that out data is in good condition to move forward with our tasks.

Identifying and Removing Outliers

Identifying and removing outliers is the second step in this data pre-processing process. This includes finding the outliers which are defined as the values which are far from extreme values of a particular attribute and make no sense in the data. These values and rows will hamper the model in learning the data set and predict the future values.
Outliers are found out by plotting boxplots of the attributes.
The Code to identify the outliers is as follows:



```
In [124…
outlier_plot_1=sns.boxplot(data=df,y='temp')
plt.show()
outlier_plot_2=sns.boxplot(data=df,y='atemp')
plt.show()
outlier_plot_3=sns.boxplot(data=df,y='windspeed')
plt.show()
outlier_plot_4=sns.boxplot(data=df,y='humidity')
plt.show()
```

Fig 3.

The results and the plot from the above code is as follows:

1. Temperature attribute: No outliers found here

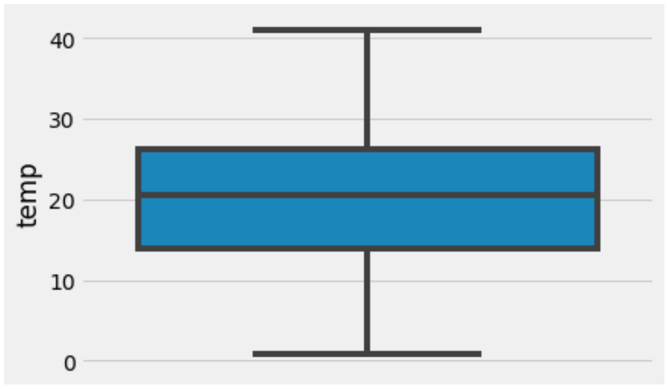


Fig 4.

2. Apparent Temperature: No outliers found here

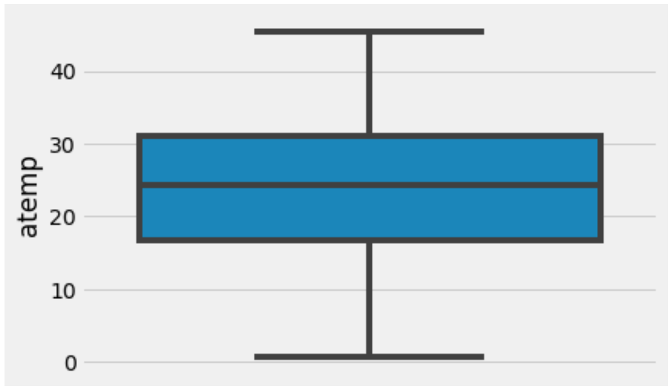


Fig 5.

3. Windspeed: Outliers are present here and are present above the value of 30

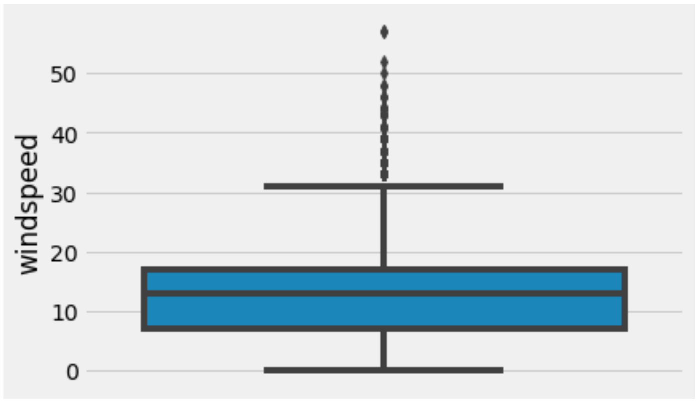


Fig 6.

4. Humidity: Outliers can be seen here and below the value of 20

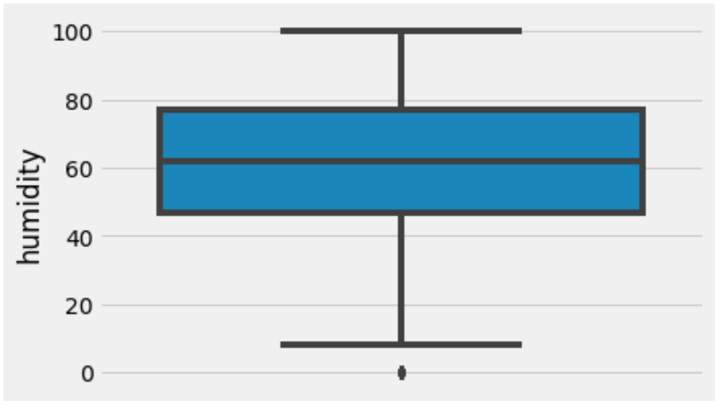
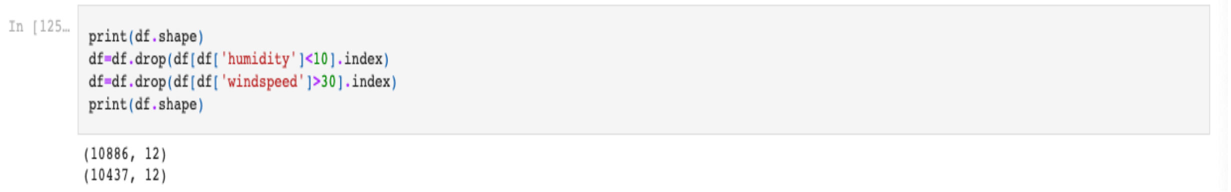


Fig 7.

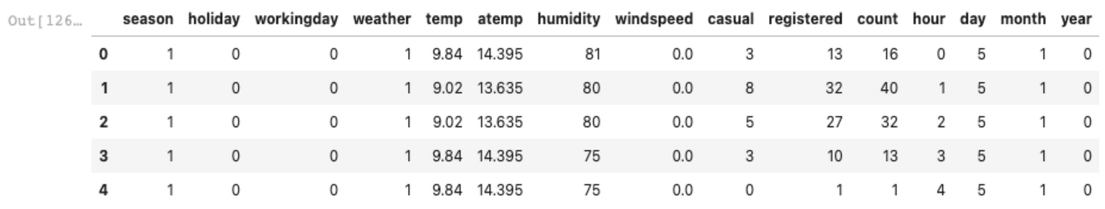
Removing the Outliers, we use the following code and also we need to verify that these rows which are outliers are removed and hence we take a look at the dimensions of the dataframe which can help us in comparing the rows.



```
In [125…
print(df.shape)
df=df.drop(df[df['humidity']<10].index)
df=df.drop(df[df['windspeed']>30].index)
print(df.shape)

(10886, 12)
(10437, 12)
```

Fig 8.

Next step we take in the data pre-processing is converting and decoding some variables in order to make more data and attributes relevant for the model. Here we had a attribute named ‘datetime’ in the original dataset which can be split into different columns like ‘day’, ’hour’, ’month’, ’year’.
After doing this the dataset now looks like this:



Out[126...

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	hour	day	month	year
0	1	0	0	1	9.84	14.395	81	0.0	3	13	16	0	5	1	0
1	1	0	0	1	9.02	13.635	80	0.0	8	32	40	1	5	1	0
2	1	0	0	1	9.02	13.635	80	0.0	5	27	32	2	5	1	0
3	1	0	0	1	9.84	14.395	75	0.0	3	10	13	3	5	1	0
4	1	0	0	1	9.84	14.395	75	0.0	0	1	1	4	5	1	0

Fig 9.

Visualizations

After the data -pre-processing stage is accomplished and completed we move towards the visualizations stage. Here we plot some interesting plots and dig deeply into the plots to understand the relations between different attributes and how these can be interpreted and what all information can these give.

Plot 1: Number of bike rentals during a particular weather

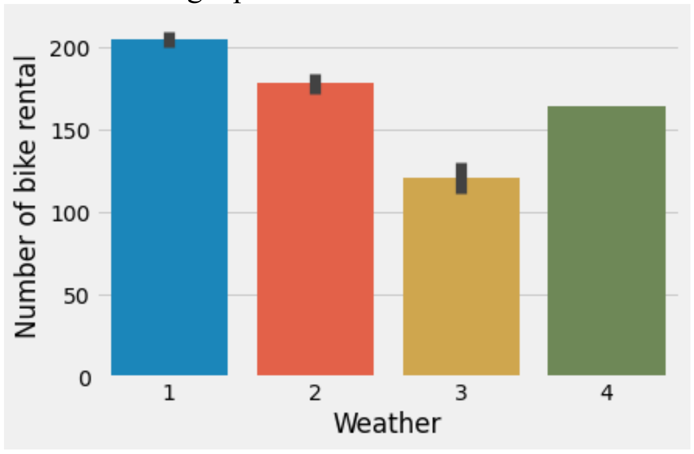


Fig 10.

We can see that the maximum bikes were rented during Weather 1 which corresponds to Clear, Few clouds, partly cloudy, Partly cloudy, and least during weather 3 which is Snow.

Plot 2: Number of bike rentals for each month

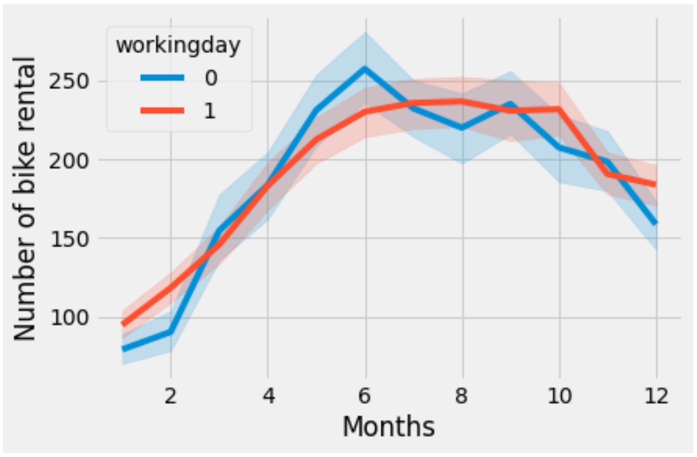


Fig 11.

We can see that in Plot 2, the Maximum bikes were rented during the period of 6-8th months for when it was a working day and near the 5th month when it was not a working day.

Plot 3: Number of Bike rentals during time of day categorized by working day:

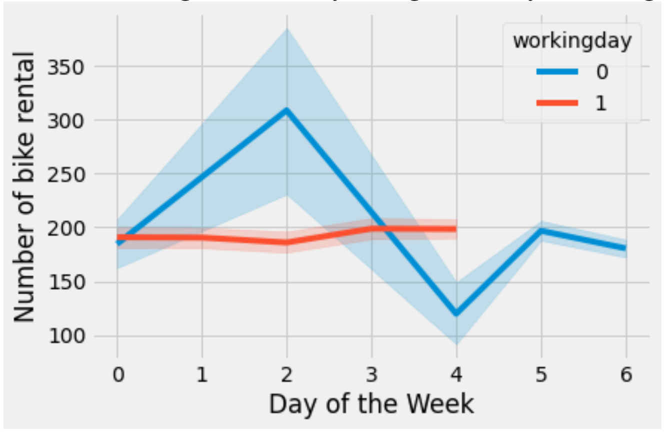


Fig 12.

We can see that the max bikes were rented during the 1st and 3rd days of the week.

Plot 4: Number of bike rentals with the humidity on different days categorized by holiday:

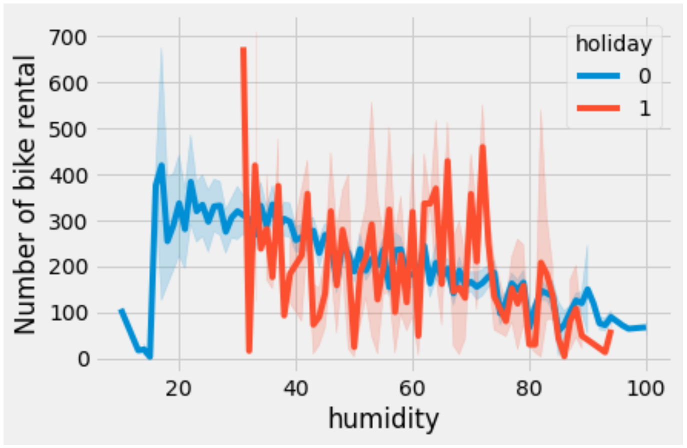


Fig 13.

It can be seen that the bikes rentals have been very fluctuating in this graph but one trend that can be seen is as follows that as humidity increases the bike rentals decreases.

Plot 5: Number of bike rentals versus the temperatures during the different days:

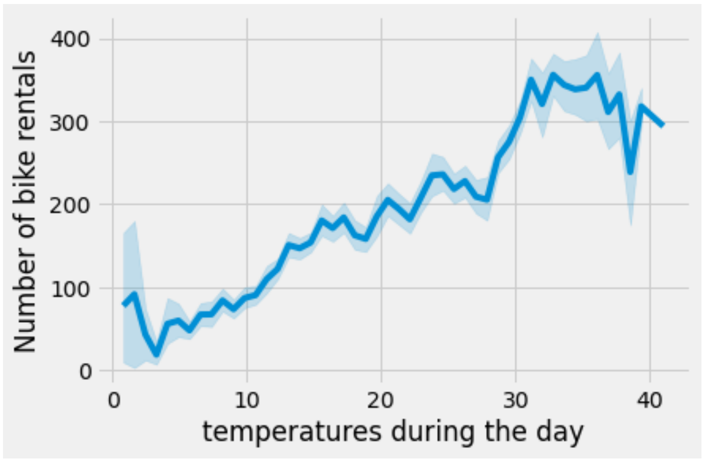


Fig 14.

When the temperatures have risen the bike rentals have also risen.

Plot 6: Number of Bike rentals by seasons:

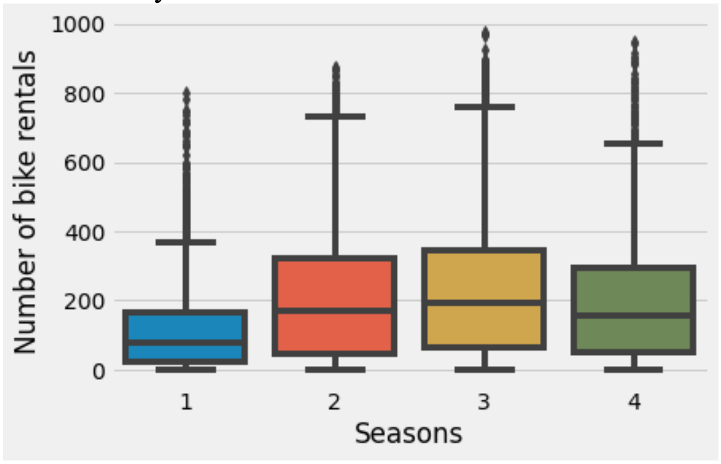


Fig 15.

Most bikes were rented in the third Season and the least in the first season which are the cold months.

Plot 7: Number of bike rentals with windspeed during different days:

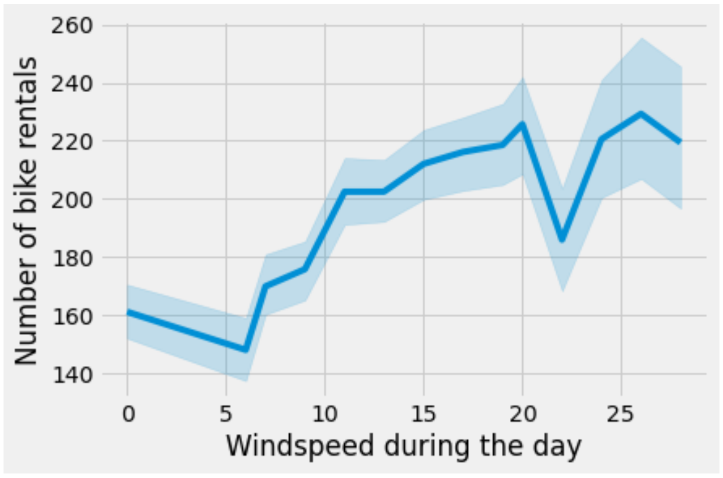
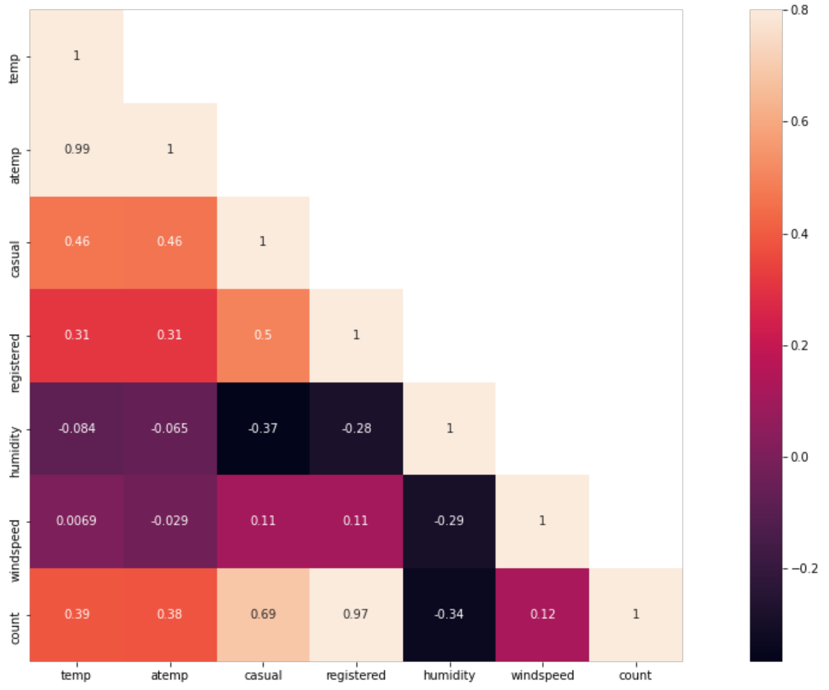


Fig 16.

We can see from the above graph that the as the windspeed have increases over the value of 5, the rentals have increased till 17 and then slightly decrease and then increased again.

Visualizing the correlation of the different variables with each other
We need to find out how relevant the attributes are and how they influence each other. Hence, we plot a heatmap using the seaborn package. The heatmap is as follows:

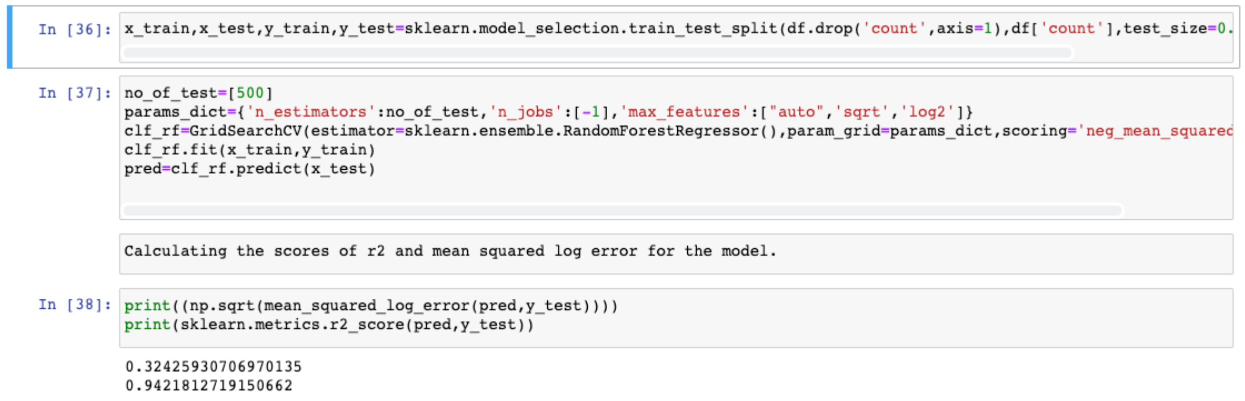


	temp	atemp	casual	registered	humidity	windspeed	count
temp	1						
atemp	0.99	1					
casual	0.46	0.46	1				
registered	0.31	0.31	0.5	1			
humidity	-0.084	-0.065	-0.37	-0.28	1		
windspeed	0.0069	-0.029	0.11	0.11	-0.29	1	
count	0.39	0.38	0.69	0.97	-0.34	0.12	1

Fig 17.

Model Selection and Training
We have already found the influence of each attribute on each other and now it's time to select a model for supervised and unsupervised learning.

For supervised learning, after comparing the RMSE scores, Random Forest Regression have the best scores and the accuracy to train a model and predict the values.
We will be Predicting the ‘count’ Variable with this regression technique.
Also, we find the mean squared error and the R2 Square for the model.
These scores evaluated how efficient and what is the error margin that this model can provide us with.
The code for this is as follows:



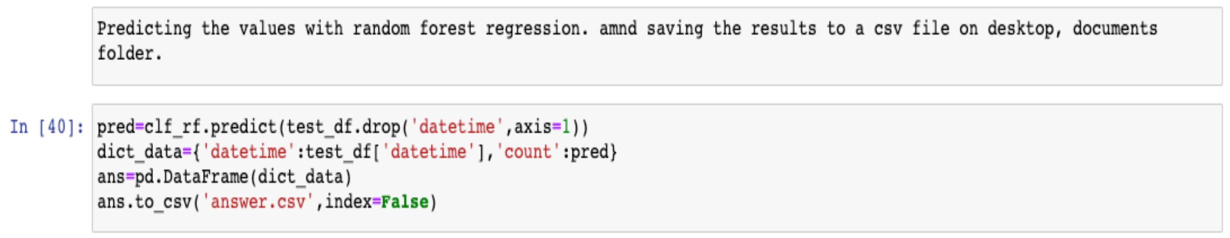
```
In [36]: x_train,x_test,y_train,y_test=sklearn.model_selection.train_test_split(df.drop('count',axis=1),df['count'],test_size=0.

In [37]: no_of_test=[500]
         params_dict={'n_estimators':no_of_test,'n_jobs':[-1],'max_features':["auto",'sqrt','log2']}
         clf_rf=GridSearchCV(estimator=sklearn.ensemble.RandomForestRegressor(),param_grid=params_dict,scoring='neg_mean_squarec
         clf_rf.fit(x_train,y_train)
         pred=clf_rf.predict(x_test)
```

```
Calculating the scores of r2 and mean squared log error for the model.
```

```
In [38]: print((np.sqrt(mean_squared_log_error(pred,y_test))))
         print(sklearn.metrics.r2_score(pred,y_test))

         0.32425930706970135
         0.9421812719150662
```

Fig 18.

As we can see that the mean square error for this model is 32.42% and the R2 score for the same model is 94.21% which is an acceptable value.
For the Prediction phase and the saving the result in order to compare, the code is as follows:

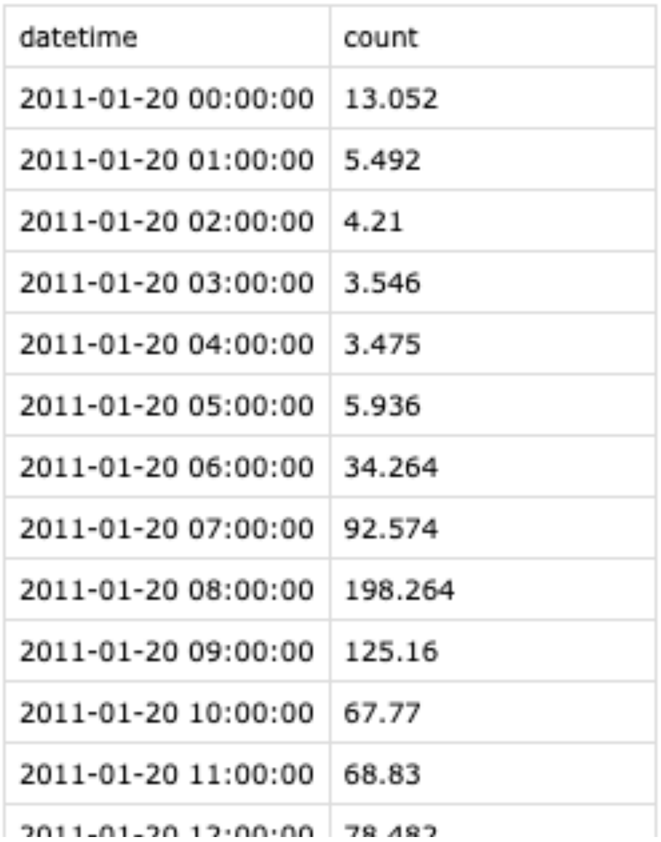


```
Predicting the values with random forest regression. amnd saving the results to a csv file on desktop, documents
folder.
```

```
In [40]: pred=clf_rf.predict(test_df.drop('datetime',axis=1))
         dict_data={'datetime':test_df['datetime'],'count':pred}
         ans=pd.DataFrame(dict_data)
         ans.to_csv('answer.csv',index=False)
```

Fig 19.

This model is used as a supervised learning model and used to predict the ‘count’ attribute and we have successfully predicted the values.
The output of the answer file looks like as follows:



datetime	count
2011-01-20 00:00:00	13.052
2011-01-20 01:00:00	5.492
2011-01-20 02:00:00	4.21
2011-01-20 03:00:00	3.546
2011-01-20 04:00:00	3.475
2011-01-20 05:00:00	5.936
2011-01-20 06:00:00	34.264
2011-01-20 07:00:00	92.574
2011-01-20 08:00:00	198.264
2011-01-20 09:00:00	125.16
2011-01-20 10:00:00	67.77
2011-01-20 11:00:00	68.83
2011-01-20 12:00:00	78.482

Fig 20.

Unsupervised Algorithm using K-Means clustering
K-Means clustering is used to cluster the data in the dataset to make different and classify them by using the clusters. This way we will have different clusters with different values for each attribute and hence, the required attribute can be predicted using the values of the attributes that can be found in a particular cluster.
This unsupervised algorithm works well when the data goes with it and there is a clear cluster which can be formed which will help the model predict the values required accurately. This dataset however is not that fine-tuned to work with the unsupervised algorithms but still we can look at some clusters forming with this algorithm.
First step is the value of the K that is the number of clusters that need to be formed. Now, this value can be arbitrary but there are certain methods to find the value. The method we are using is the elbow method, where we plot a graph between the SSE(Sum of Squared Errors) and the number of clusters. In the Graph we can look at a bend, the x-axis value of the bend point is taken as the number of clusters and the value of K.
The graph looks as follows for this dataset:

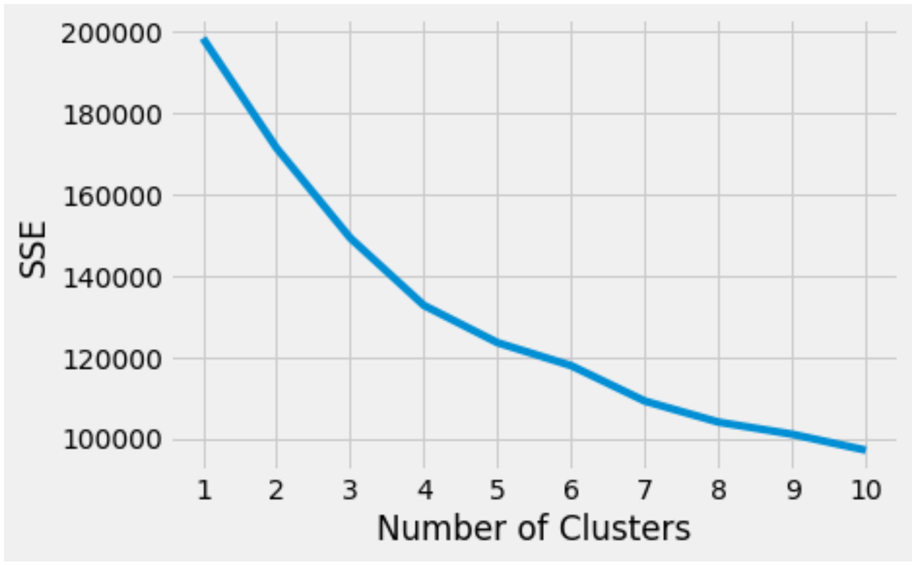
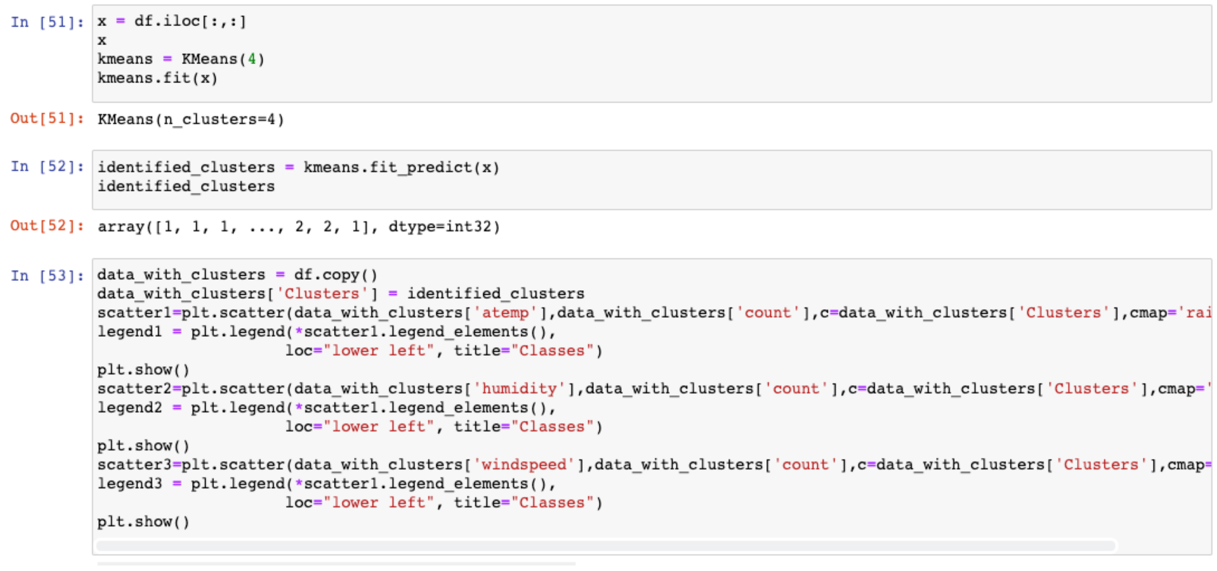


Fig 21.

Here, in the graph the bend appears at 4 hence, we take the value of k=4.

Now, the code for the above algorithms is as follows:



```
In [51]: x = df.iloc[:,:]
         x
         kmeans = KMeans(4)
         kmeans.fit(x)

Out[51]: KMeans(n_clusters=4)

In [52]: identified_clusters = kmeans.fit_predict(x)
         identified_clusters

Out[52]: array([1, 1, 1, ..., 2, 2, 1], dtype=int32)

In [53]: data_with_clusters = df.copy()
         data_with_clusters['Clusters'] = identified_clusters
         scatter1=plt.scatter(data_with_clusters['atemp'],data_with_clusters['count'],c=data_with_clusters['Clusters'],cmap='rai
         legend1 = plt.legend(*scatter1.legend_elements(),
                             loc="lower left", title="Classes")
         plt.show()
         scatter2=plt.scatter(data_with_clusters['humidity'],data_with_clusters['count'],c=data_with_clusters['Clusters'],cmap='
         legend2 = plt.legend(*scatter1.legend_elements(),
                             loc="lower left", title="Classes")
         plt.show()
         scatter3=plt.scatter(data_with_clusters['windspeed'],data_with_clusters['count'],c=data_with_clusters['Clusters'],cmap=
         legend3 = plt.legend(*scatter1.legend_elements(),
                             loc="lower left", title="Classes")
         plt.show()
```

Fig 22.

The scatterplots with the clusters that are formed is as follows:

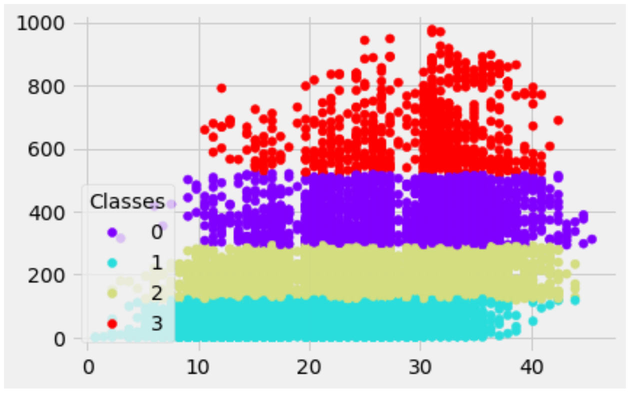


Fig 23.

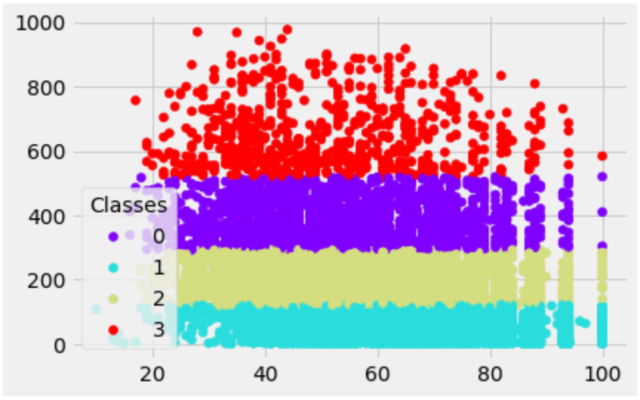


Fig 24.

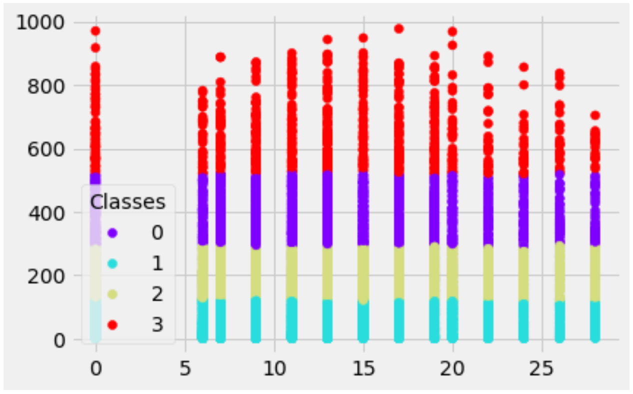
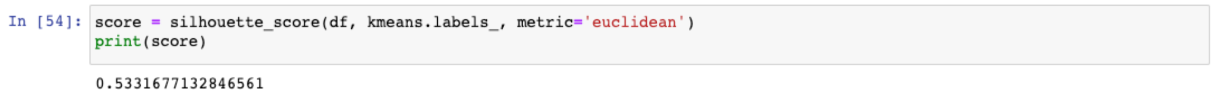


Fig 25.

These clusters will have a set of values for the different attributes and the input will then be compared to the values of these attributes, then the cluster with the least distance, either Manhattan distance or direct distance3will be selected as the favorable cluster.

For scoring this K-Means clustering, we use silhouette score and a Euclidean metric. Silhouette score for a set of sample data points is used to measure how dense and well-separated the clusters are. The silhouette score falls within the range [-1, 1]. The silhouette score of 1 means that the clusters are very dense and nicely separated. The score of 0 means that clusters are overlapping. The score of less than 0 means that data belonging to clusters may be wrong/incorrect.

The code for the Silhouette scoring is as follows:



```
In [54]: score = silhouette_score(df, kmeans.labels_, metric='euclidean')
         print(score)

0.5331677132846561
```

Fig 26.

As we can see that the score is 0.533 which means that the clusters are favorable and are densely populated and fairly well separated. This score is favorable in clustering different attributes and hence can be used to classify conditions.

Conclusion
From the data that we have been using, we have successfully predicted the bikes rented using the Random Forest Regression technique. We have an efficiency of 94.19 % with this model which is a good score to predict the needed variable.

We have also successfully classified different clusters using an unsupervised learning algorithm which is the K-Means clustering and successfully identified clusters using the model.

References
1. https://www.kaggle.com/c/bike-sharing-demand/data : Kaggle URL for Data information
2. https://dzone.com/articles/kmeans-silhouette-score-explained-with-python-exam : K-Means Clustering
3. https://pandas.pydata.org/pandas-docs/stable/reference/frame.html : Pandas Documentation
4. https://scikit-learn.org/stable/index.html : Scikit-Learn for modelling documentation for python
5. https://matplotlib.org/stable/index.html : Matplotlib documentation