Homework 01

In [1]:

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import r2_score
```

In [2]:

```
#Reading the csv file and dropping the feature instant which is unnecessary as i
ts only a serial number.
mydata = pd.read_csv('hour.csv')
mydata = mydata.drop('instant',axis = 1)
mydata
```

Out[2]:

	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atem
0	2011-01-01	1	0	1	0	0	6	0	1	0.24	0.287
1	2011-01-01	1	0	1	1	0	6	0	1	0.22	0.272
2	2011-01-01	1	0	1	2	0	6	0	1	0.22	0.272
3	2011-01-01	1	0	1	3	0	6	0	1	0.24	0.287
4	2011-01-01	1	0	1	4	0	6	0	1	0.24	0.287
...	...	...	...	...	...	...	...	...	...	...	
17374	2012-12-31	1	1	12	19	0	1	1	2	0.26	0.257
17375	2012-12-31	1	1	12	20	0	1	1	2	0.26	0.257
17376	2012-12-31	1	1	12	21	0	1	1	1	0.26	0.257
17377	2012-12-31	1	1	12	22	0	1	1	1	0.26	0.272
17378	2012-12-31	1	1	12	23	0	1	1	1	0.26	0.272

17379 rows × 16 columns

In [8]:

```
z = mydata.drop(['dteday'],axis = 1)
a = z-z.mean()
b = a.T@a
c = b/17379
Std_z = z.std()
d = np.multiply.outer(Std_z.to_numpy(),Std_z.to_numpy())
correlation_matrix = c/d
correlation_matrix
#sns.heatmap(e)
#correlation_matrix = mydata.corr()
#correlation_matrix
#This gives us the autocorrelation matrix. It tells us how each feature is relat
ed to the other.
```

Out[8]:

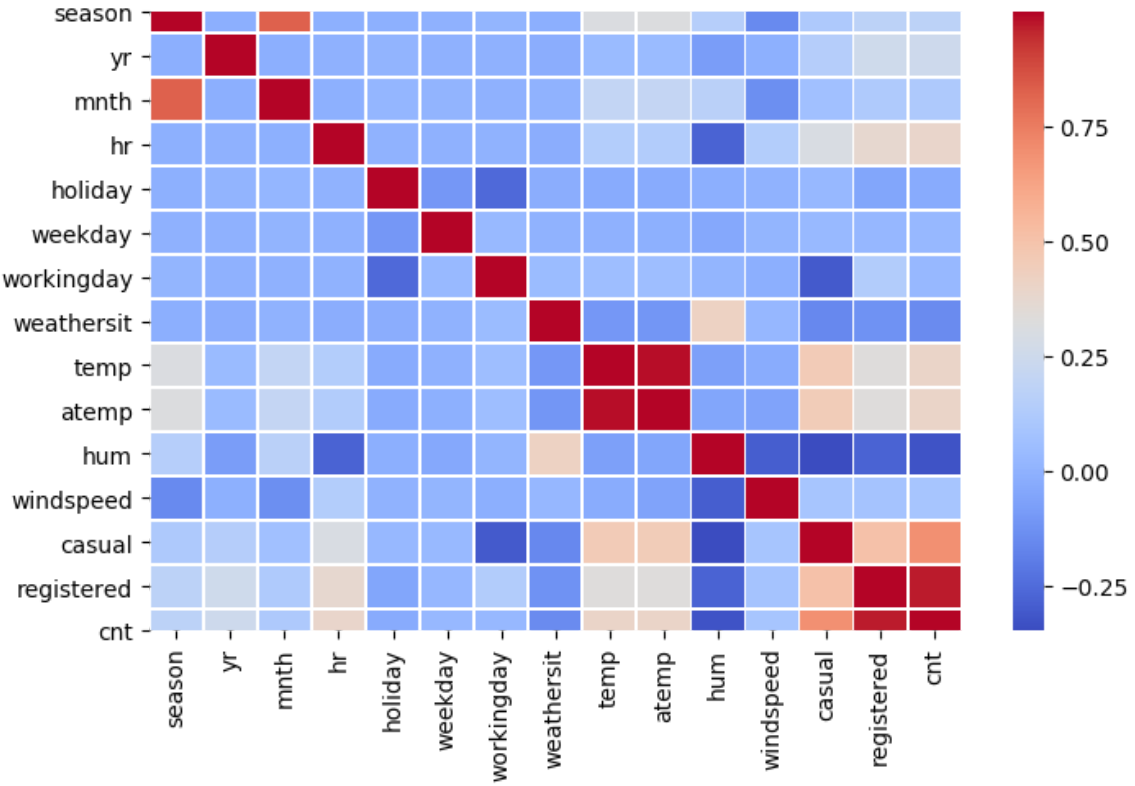
	season	yr	mnth	hr	holiday	weekday	workingday	wea
season	0.999942	-0.010742	0.830338	-0.006117	-0.009584	-0.002335	0.013742	-0.(
yr	-0.010742	0.999942	-0.010472	-0.003867	0.006691	-0.004485	-0.002196	-0.(
mnth	0.830338	-0.010472	0.999942	-0.005772	0.018429	0.010399	-0.003477	0.(
hr	-0.006117	-0.003867	-0.005772	0.999942	0.000479	-0.003498	0.002285	-0.(
holiday	-0.009584	0.006691	0.018429	0.000479	0.999942	-0.102082	-0.252457	-0.(
weekday	-0.002335	-0.004485	0.010399	-0.003498	-0.102082	0.999942	0.035953	0.(
workingday	0.013742	-0.002196	-0.003477	0.002285	-0.252457	0.035953	0.999942	0.(
weathersit	-0.014523	-0.019156	0.005399	-0.020201	-0.017035	0.003311	0.044670	0.9
temp	0.312007	0.040911	0.201680	0.137596	-0.027339	-0.001795	0.055387	-0.1
atemp	0.319361	0.039219	0.208084	0.133742	-0.030971	-0.008820	0.054664	-0.1
hum	0.150616	-0.083542	0.164402	-0.276482	-0.010588	-0.037156	0.015687	0.4
windspeed	-0.149764	-0.008739	-0.135379	0.137244	0.003987	0.011501	-0.011829	0.(
casual	0.120200	0.142770	0.068453	0.301184	0.031562	0.032720	-0.300925	-0.1
registered	0.174216	0.253670	0.122266	0.374119	-0.047343	0.021577	0.134318	-0.1
cnt	0.178045	0.250480	0.120631	0.394049	-0.030926	0.026898	0.030283	-0.1

In [9]:

```
figure = plt.figure(figsize = (8,5),dpi = 100)
sns.heatmap(correlation_matrix, linewidths=.5,cmap = 'coolwarm')
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a27020f10>
```



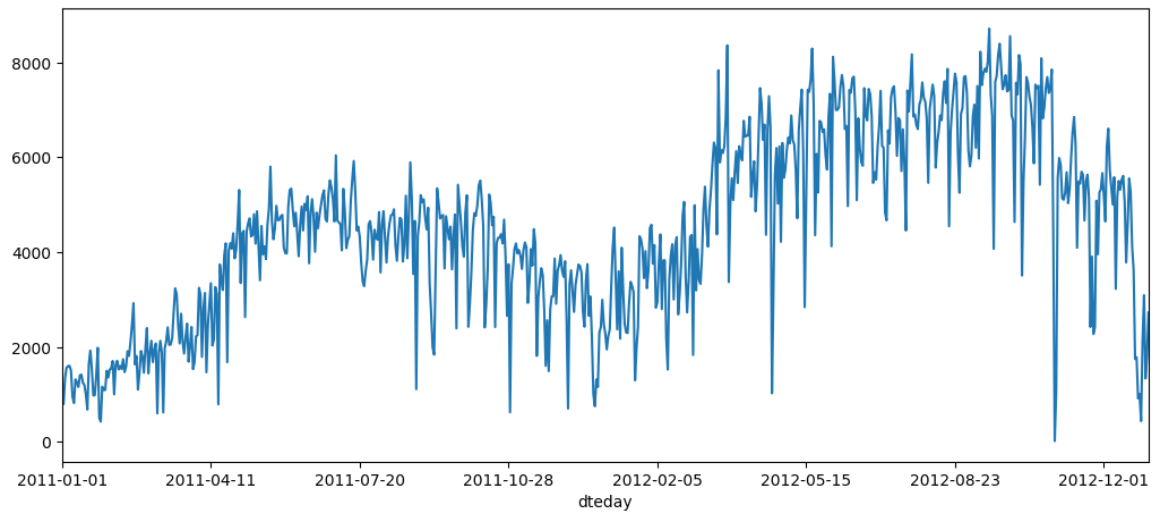
Exploratory Analysis

In [162]:

```
#Date VS Cnt
#Here i have grouped the counts per date's such that all the bicycles on 2011-01
-01 have been summed up and printed.
fig = plt.figure(figsize = (12,5),dpi =100)
axes = fig.add_axes([1,0.1,0.8,0.8])
mydata.groupby('dteday').sum()['cnt'].plot()

#Observation: I feel the per day data is very non informative and doesnt give us
a trend to predict the cnt
```

Out[162]:

```
<matplotlib.axes._axes.Axes at 0x1a3e820c90>
```

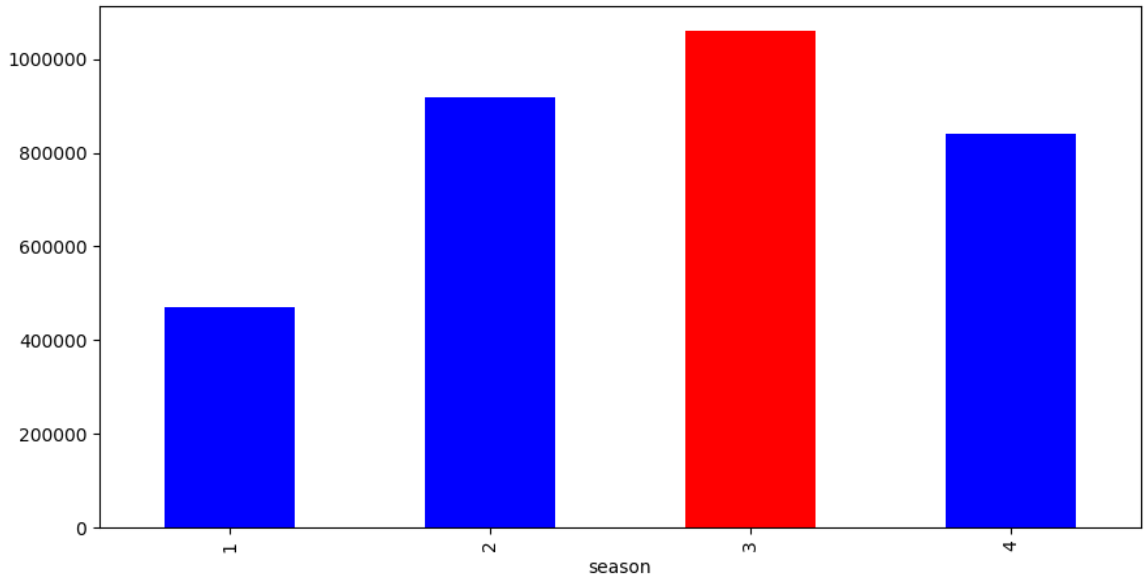


In [103]:

```
#Season VS Cnt
#Here i have grouped the seasons  such that all the bicycles in spring have been
summed up and plotted.
fig = plt.figure(figsize = (10,5),dpi =100)
axes = fig.add_axes([1,0.1,0.8,0.8])
mydata.groupby('season').sum()['cnt'].plot(kind = 'bar', color = ['b','b','r',
'b'])

#Observation: We can tell that the count is highest in the fall season and least
in spring season
#We can also drop this feature as it  is highly correlated with month feature wh
ich basically tells us the same information.
```

Out[103]:

```
<matplotlib.axes._axes.Axes at 0x1a2869de90>
```

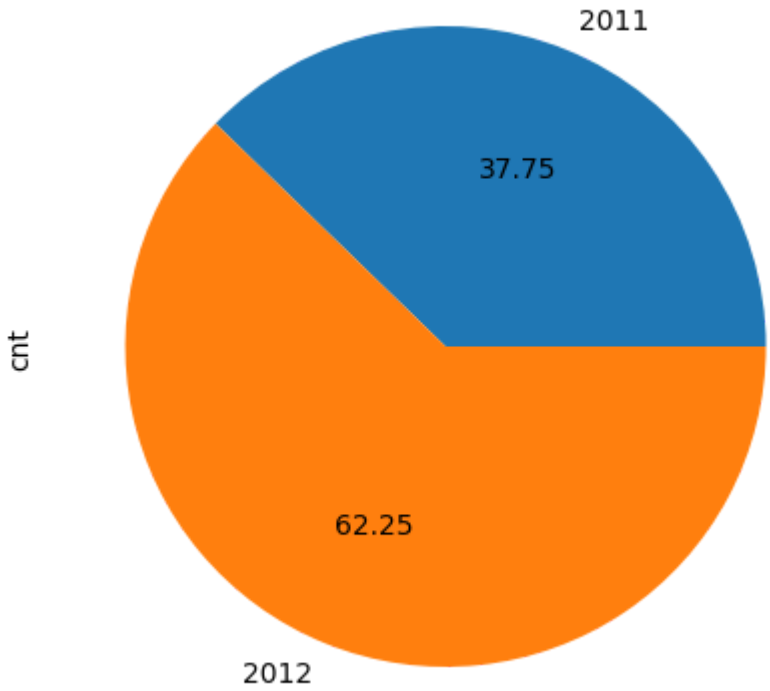


In [100]:

```
#Year VS Cnt
#Here i have grouped the Years  such that all the bicycles in 2011 have been sum
med up and plotted.
fig = plt.figure(figsize = (10,5),dpi =100)
axes = fig.add_axes([1,0.1,0.8,0.8])
mydata.groupby('yr').sum()['cnt'].plot(kind = 'pie',labels=['2011','2012'],autop
ct='%.2f')

#Observation: We can tell that the count is highest in the 2012 year and least i
n 2011
```

Out[100]:

```
<matplotlib.axes._axes.Axes at 0x1a27dc2450>
```

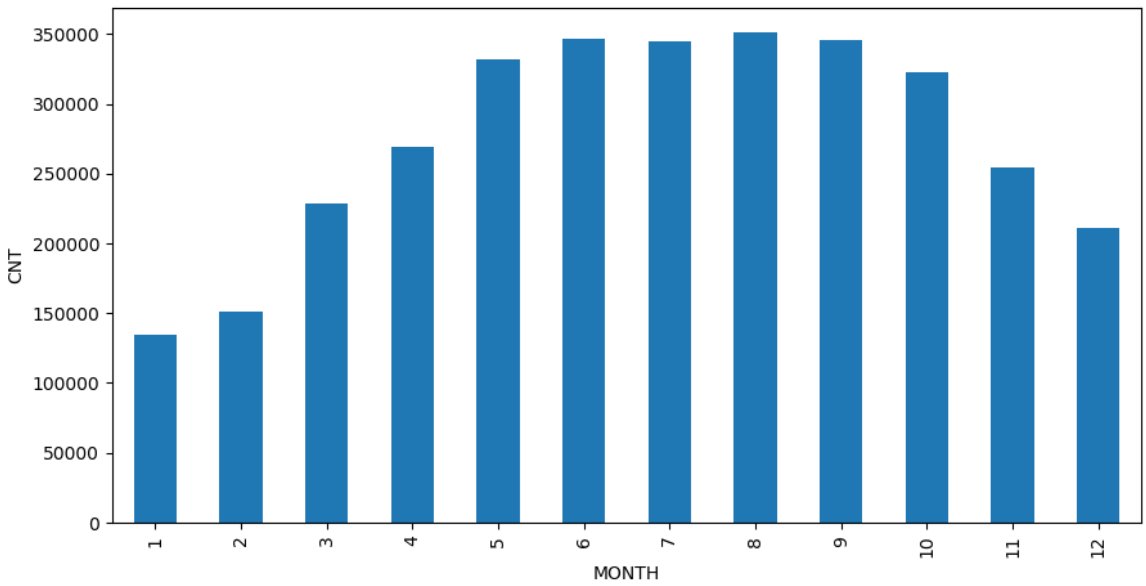


In [108]:

```
#Month VS Cnt
#Here i have grouped the Months such that all the bicycles in december have been
summed up and plotted.
fig = plt.figure(figsize = (10,5),dpi =100)
axes = fig.add_axes([1,0.1,0.8,0.8])
mydata.groupby('mnth').sum()['cnt'].plot.bar()
axes.set_ylabel('CNT')
axes.set_xlabel('MONTH')
#Observation: We can tell that the count is highest in months of june july augus
t sept and decreses in jan feb nov december when the weather is relatively cold.
```

Out[108]:

```
Text(0.5, 0, 'MONTH')
```

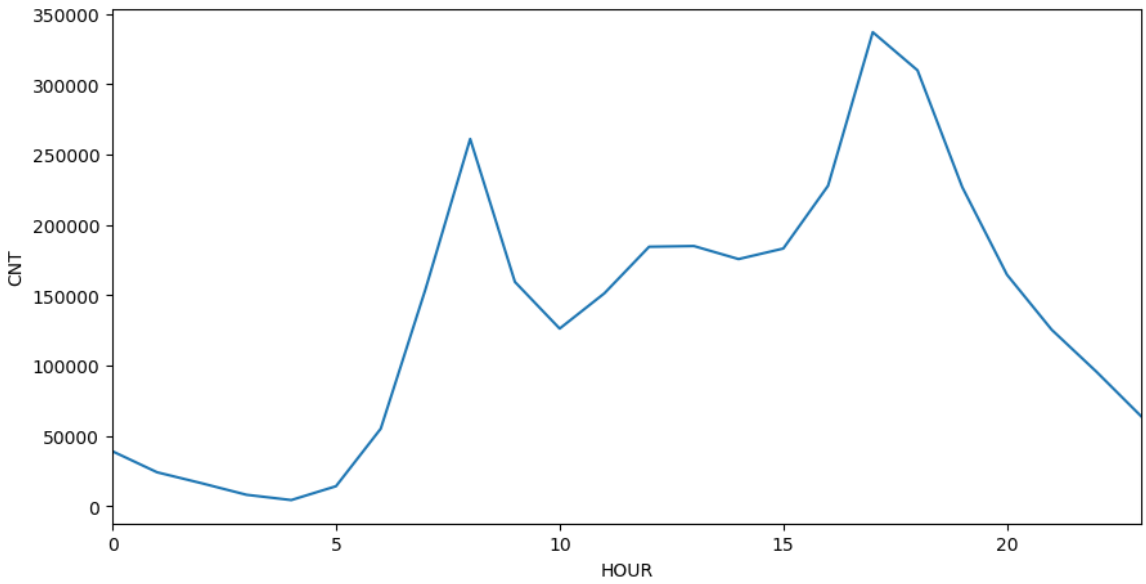


In [163]:

```
#Hour VS Cnt
#Here i have grouped the hours in every month such that all the bicycles in the
 1st hour have been summed up and plotted.
fig = plt.figure(figsize = (10,5),dpi =100)
axes = fig.add_axes([0.1,0.1,0.8,0.8])
mydata.groupby('hr').sum()['cnt'].plot.line()
axes.set_ylabel('CNT')
axes.set_xlabel('HOUR')
#Observation: We can tell that the count is highest in the hours of 9am which is
probably due to school and office start and during the noon time when people lea
ve work.
```

Out[163]:

```
Text(0.5, 0, 'HOUR')
```

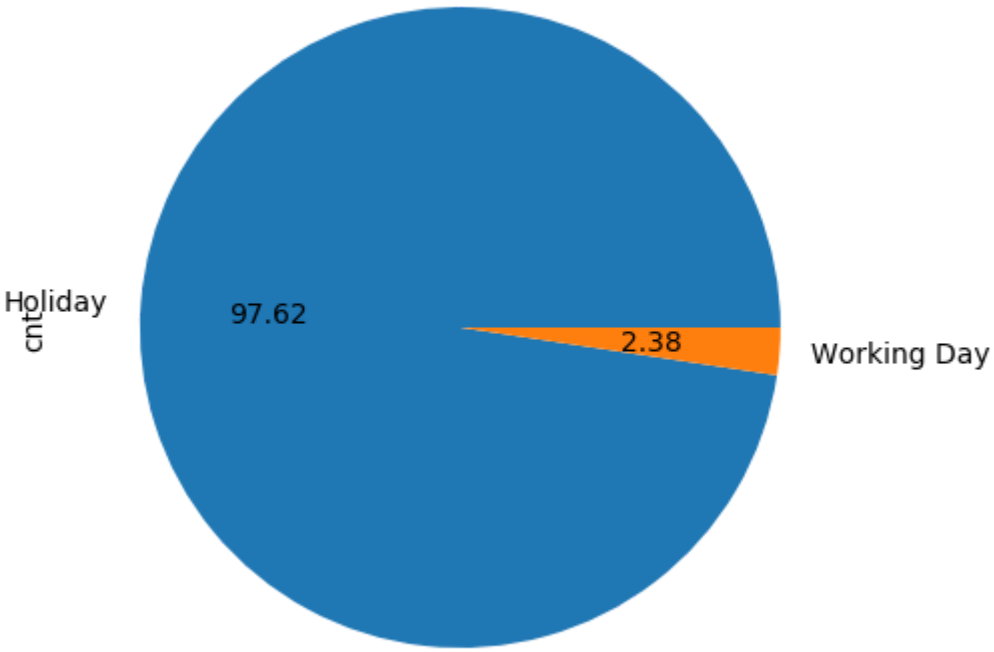


In [120]:

```
#Holiday VS Cnt
#Here i have grouped the holidays such that all the bicycles rented on holidsays
have been summed up and plotted.
fig = plt.figure(figsize = (10,5),dpi =100)
axes = fig.add_axes([1,0.1,0.8,0.8])
mydata.groupby('holiday').sum()['cnt'].plot(kind = 'pie',labels = ['Holiday','Wo
rking Day'], autopct = '%.2f')
#Observation: We can tell that the count is highest on holidays and very less on
working days.
```

Out[120]:

```
<matplotlib.axes._axes.Axes at 0x1a2a6a3750>
```

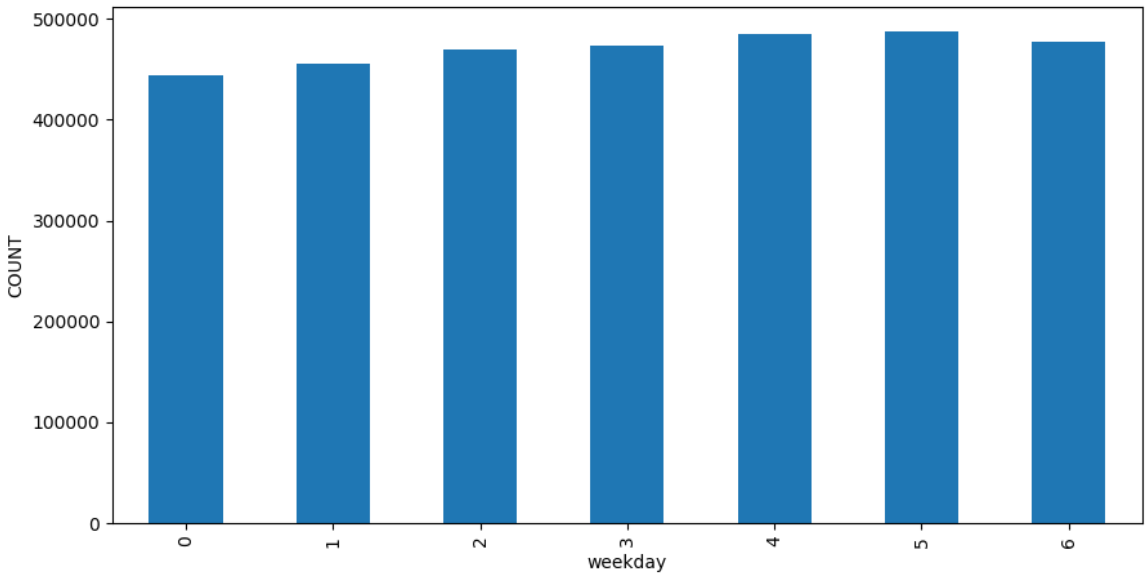


In [124]:

```
#Weekday VS Cnt
#Here i have grouped the weekdays such that all the bicycles on sat sun friday h
ave been summed up and plotted.
fig = plt.figure(figsize = (10,5),dpi =100)
axes = fig.add_axes([1,0.1,0.8,0.8])
mydata.groupby('weekday').sum()['cnt'].plot(kind = 'bar')
axes.set_ylabel('COUNT')
#Observation: We can tell that the count is pretty similar on all days.
```

Out[124]:

```
Text(0, 0.5, 'COUNT')
```

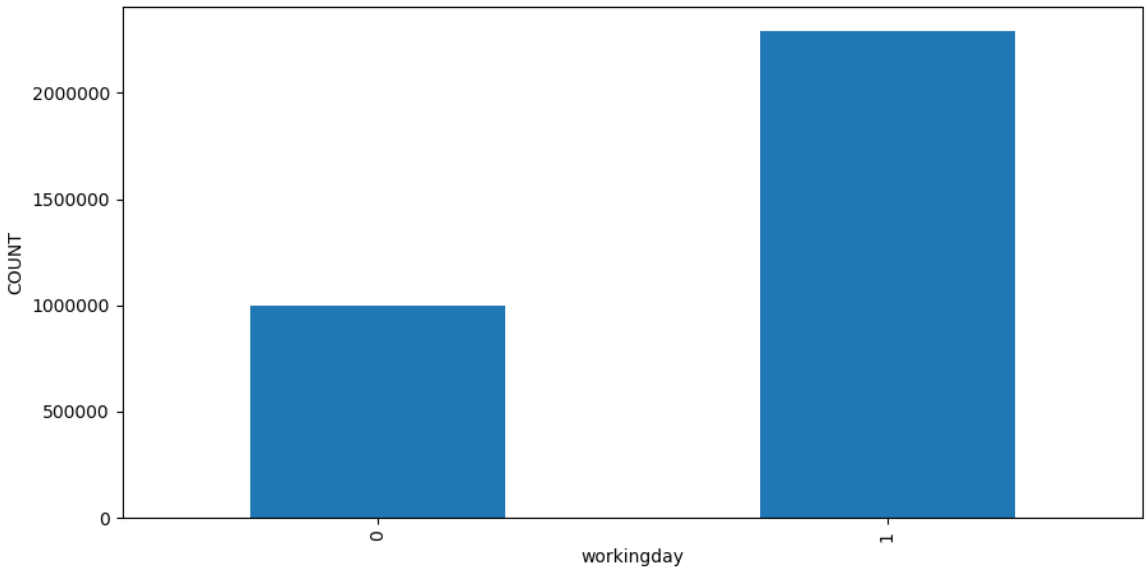


In [126]:

```
#Working day VS Cnt
#Here i have grouped the workingdays such that all the bicycles on non holidays
 have been summed up and plotted.
fig = plt.figure(figsize = (10,5),dpi =100)
axes = fig.add_axes([1,0.1,0.8,0.8])
mydata.groupby('workingday').sum()['cnt'].plot(kind = 'bar')
axes.set_ylabel('COUNT')
#Observation: We can tell that the count is very high on non holidays.
```

Out[126]:

```
Text(0, 0.5, 'COUNT')
```

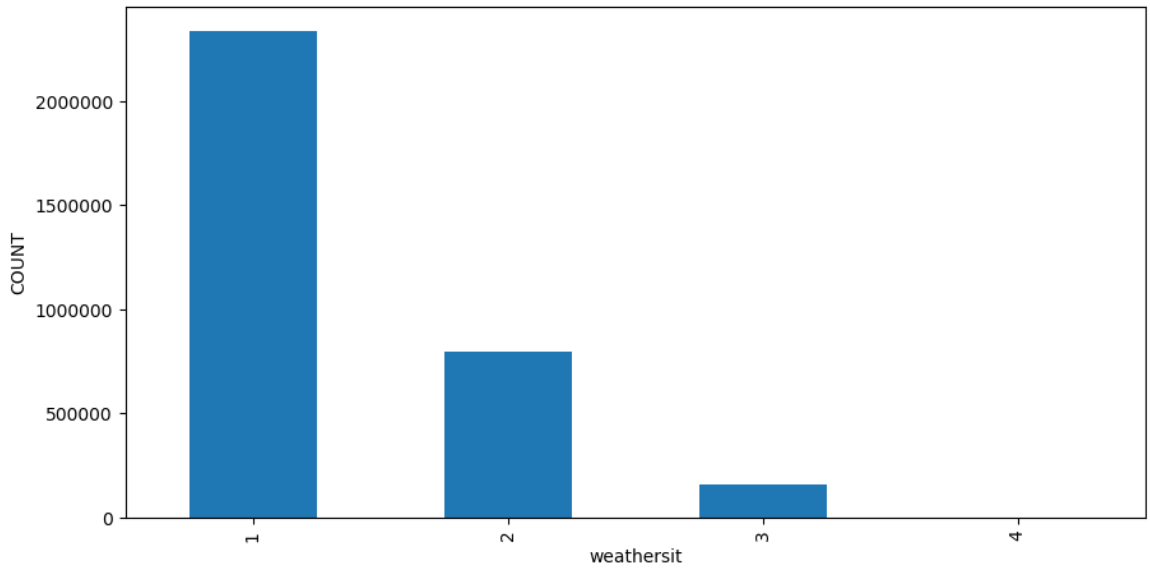


In [127]:

```
#Weather VS Cnt
#Here i have grouped the weathers such that all the bicycles on fall..summer etc
have been summed up and plotted.
fig = plt.figure(figsize = (10,5),dpi =100)
axes = fig.add_axes([1,0.1,0.8,0.8])
mydata.groupby('weathersit').sum()['cnt'].plot(kind = 'bar')
axes.set_ylabel('COUNT')
#Observation: We can tell that the count is very high on clear sky days while ve
ry less on rainy days
```

Out[127]:

```
Text(0, 0.5, 'COUNT')
```

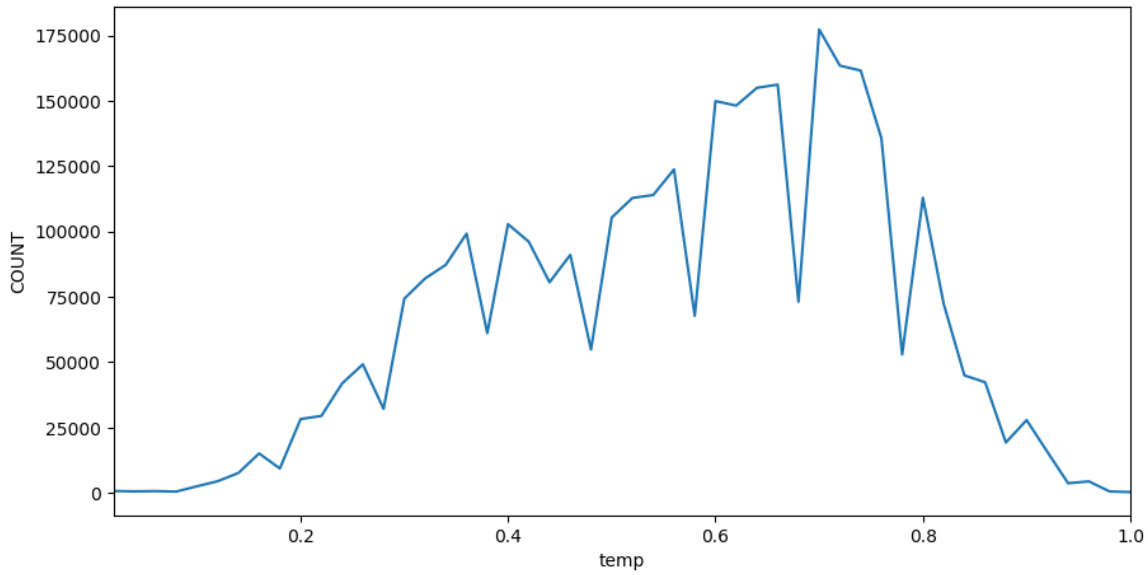


In [130]:

```
#Temp VS Cnt
#Here i have grouped the temperature such that all the bicycles on days with a c
ertain temp have been summed up and plotted.
fig = plt.figure(figsize = (10,5),dpi =100)
axes = fig.add_axes([1,0.1,0.8,0.8])
mydata.groupby('temp').sum()['cnt'].plot(kind = 'line')
axes.set_ylabel('COUNT')
#Observation: We can tell that the count is very high on days with moderate temp
and less on very hot and very cold days.
```

Out[130]:

```
Text(0, 0.5, 'COUNT')
```

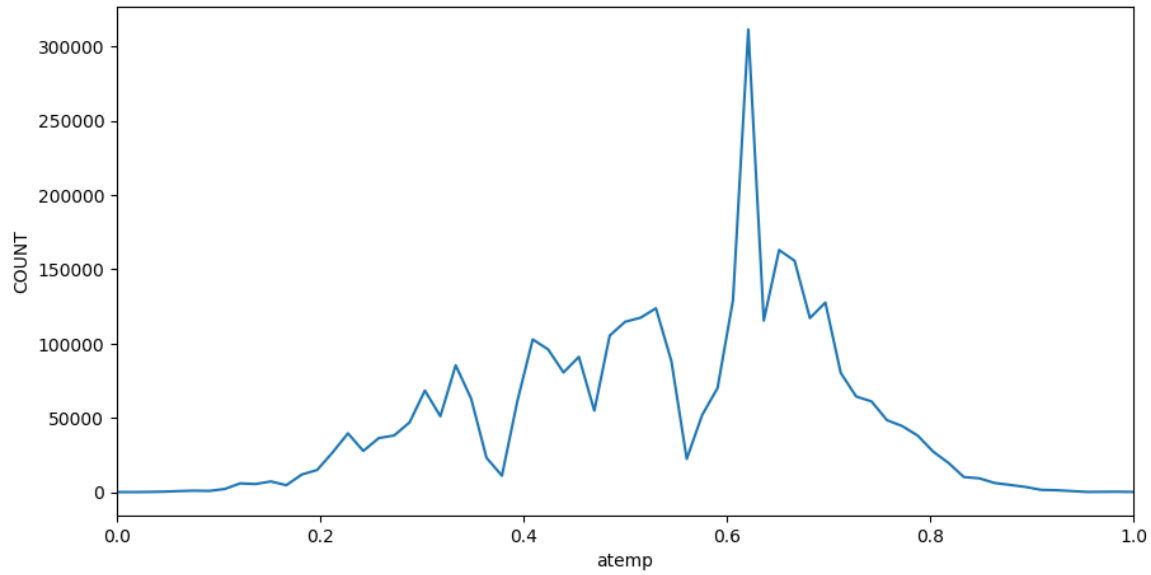


In [131]:

```
#Atemp VS Cnt
#Here i have grouped temperature such that all the bicycles on days with a certa
in temp have been summed up and plotted.
fig = plt.figure(figsize = (10,5),dpi =100)
axes = fig.add_axes([1,0.1,0.8,0.8])
mydata.groupby('atemp').sum()['cnt'].plot(kind = 'line')
axes.set_ylabel('COUNT')
#Observation: We can tell that the count is very high on days with moderate feel
ing temp and less on very hot and very cold days.
```

Out[131]:

```
Text(0, 0.5, 'COUNT')
```

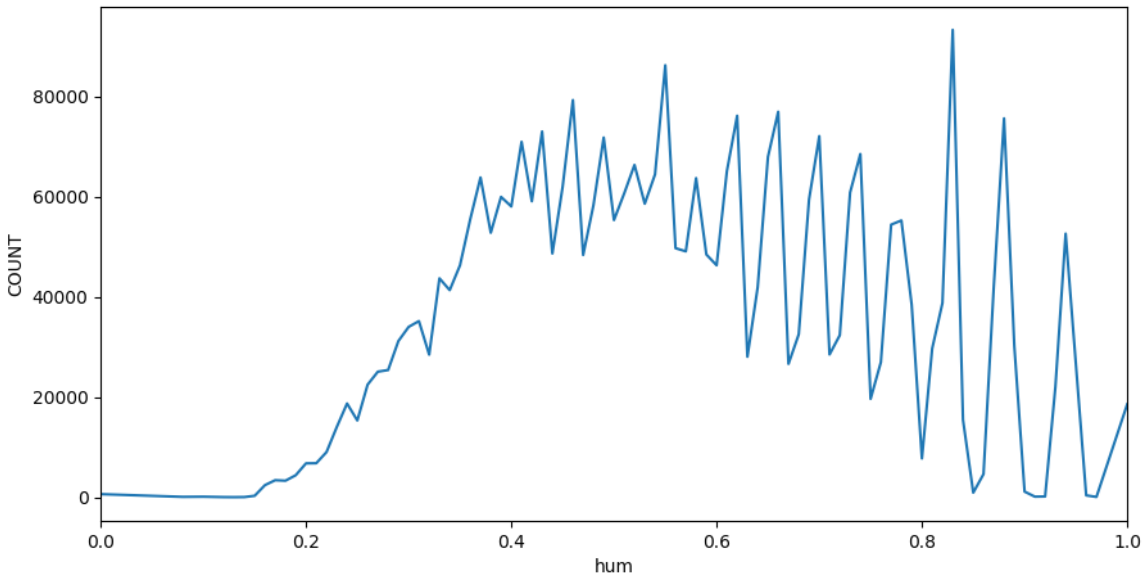


In [142]:

```
#Humidity VS Cnt
#Here i have grouped the humidity such that all the bicycles on days with xyz hu
midity have been summed up and plotted.
fig = plt.figure(figsize = (10,5),dpi =100)
axes = fig.add_axes([1,0.1,0.8,0.8])
mydata.groupby('hum').sum()['cnt'].plot()
axes.set_ylabel('COUNT')
#Observation: We can tell that the count is  high on days with moderate humidity
and less on very hot days.
```

Out[142]:

```
Text(0, 0.5, 'COUNT')
```

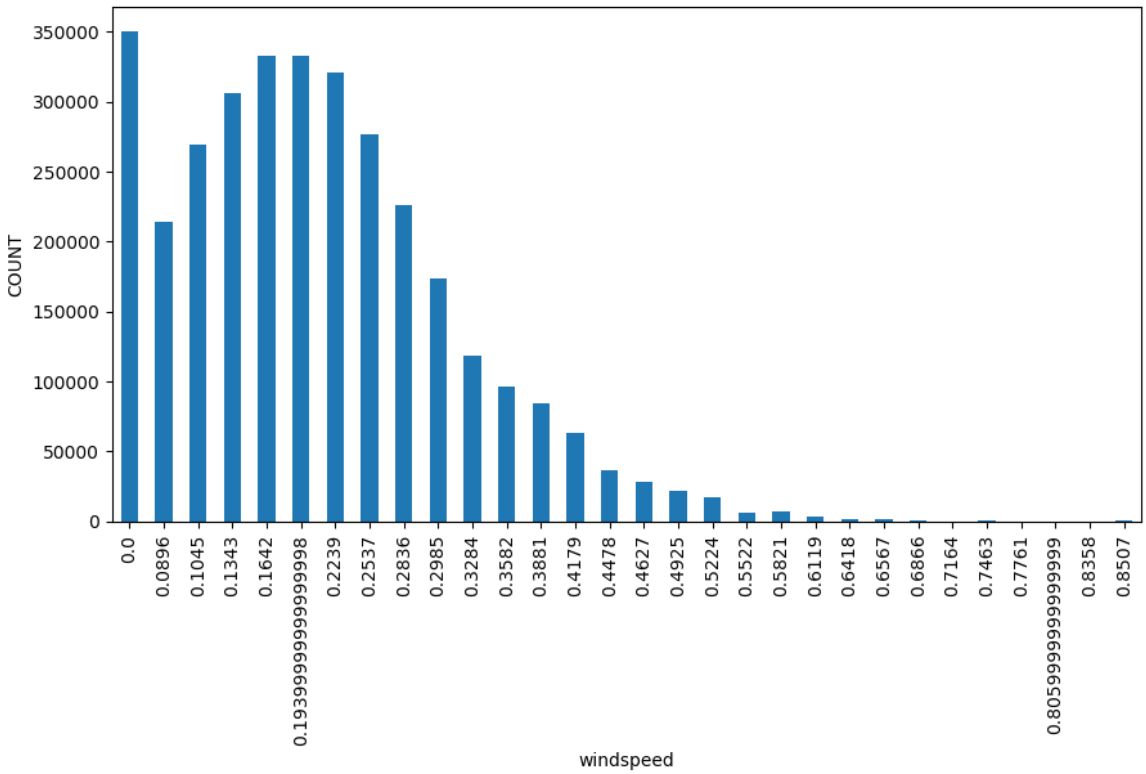


In [144]:

```
#Windspeed VS Cnt
#Here i have grouped the windy day usage such that all the bicycles on windy day
s have been summed up and plotted.
fig = plt.figure(figsize = (10,5),dpi =100)
axes = fig.add_axes([1,0.1,0.8,0.8])
mydata.groupby('windspeed').sum()['cnt'].plot.bar()
axes.set_ylabel('COUNT')
#Observation: We can tell that the count is  high on days with moderate to less
 speeds and less on very windy days.
```

Out[144]:

```
Text(0, 0.5, 'COUNT')
```

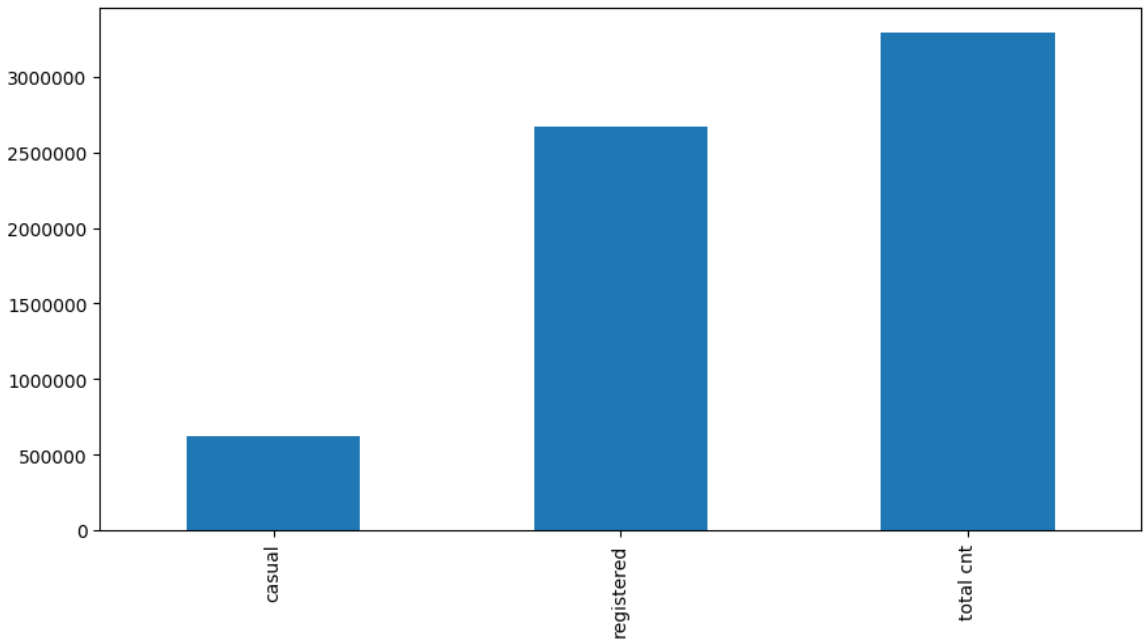


In [207]:

```
#Casual users and Registered users VS Cnt
#Here i have summed up the total casual users and the total registered users and
plotted them against the count
fig = plt.figure(figsize = (10,5),dpi =100)
axes = fig.add_axes([1,0.1,0.8,0.8])

temp = pd.DataFrame([620017,2672662,3292679],['casual','registered','total cnt'
],['number'])
temp['number'].plot(kind = 'bar')

#Observation: I summed up the total casual users and total registered users and
 we plotted it against the total cnt, we observe that we dont need both the feat
ures as theu are complimentary of each other and the data over fits if we use bo
th.. hence i dropped casual users as registered has more points.
```

Out[207]:

```
<matplotlib.axes._axes.Axes at 0x1a44d06b10>
```



Preprocessing

After studying the exploratory graphs i feel the date information is redundant. As we can see from the graph, there isnt any clear correlation between count and date.

Another field that i feel is redundant is weekday. We notice that the cnts for each day of the week is almost the same and hence it doesnt give us any useful information. We can transform this feature into weekend and weekday and observe if friday/sat/sunday affects the cnt of bikes

Also the normalised feeling temperature doesnt give us any extra information that the other temperature field hasnt already. Hence it might be redundant to our use.

We can change date feature to week feature if we wanted to.

Model Representation and Estimation

In [347]:

```
#Removing all the unnecesary features that do not help in predicting anything.

t = mydata['cnt']
cleaned_mydata = mydata.drop(['dteday','atemp','weekday','cnt','casual','season'
],axis = 1)
cleaned_mydata
```

Out[347]:

	yr	mnth	hr	holiday	workingday	weathersit	temp	hum	windspeed	registered
0	0	1	0	0	0	1	0.24	0.81	0.0000	13
1	0	1	1	0	0	1	0.22	0.80	0.0000	32
2	0	1	2	0	0	1	0.22	0.80	0.0000	27
3	0	1	3	0	0	1	0.24	0.75	0.0000	10
4	0	1	4	0	0	1	0.24	0.75	0.0000	1
...	...	...	...	...	...	...	...	...	...	...
17374	1	12	19	0	1	2	0.26	0.60	0.1642	108
17375	1	12	20	0	1	2	0.26	0.60	0.1642	81
17376	1	12	21	0	1	1	0.26	0.60	0.1642	83
17377	1	12	22	0	1	1	0.26	0.56	0.1343	48
17378	1	12	23	0	1	1	0.26	0.65	0.1343	37

17379 rows × 10 columns

In [348]:

```
#using scikit learn packages to split data randomly into sets of training and te
stign data.
from sklearn.model_selection import train_test_split
X_train,X_test,t_train,t_test = train_test_split(cleaned_mydata,t,test_size = 0.
4,random_state =101)
```

MLE

In [349]:

```
w = list(np.linalg.inv(X_train.T@X_train)@X_train.T@t_train)
print(w)
```

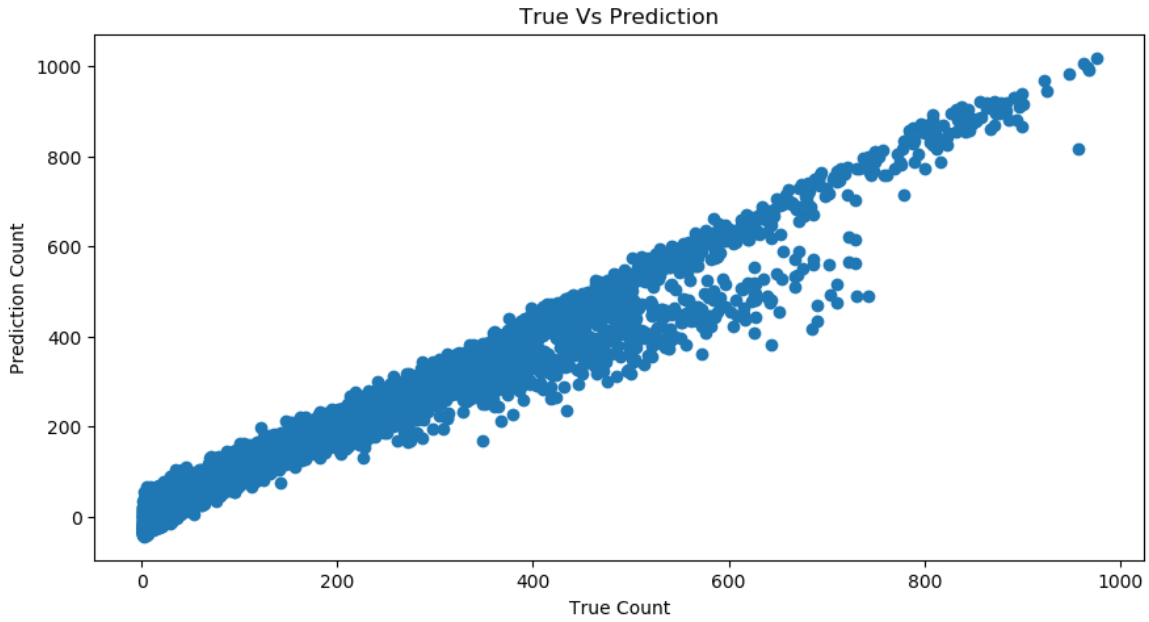
```
[3.5920038155527605, 0.09864616210592048, 0.6868847030831613, -9.480
136751071418, -37.11035515370783, 3.2361013895488355, 94.80308399578
806, -37.47015279464495, 18.553916820560584, 1.1178614750555265]
```

In [350]:

```
y = X_test@w
#X_test
fig = plt.figure(figsize = (10,5),dpi =100)
axes = fig.add_axes([1,0.1,0.8,0.8])
axes.scatter(t_test,y)
axes.set_xlabel('True Count')
axes.set_ylabel('Prediction Count')
axes.set_title('True Vs Prediction')
```

Out[350]:

```
Text(0.5, 1.0, 'True Vs Prediction')
```

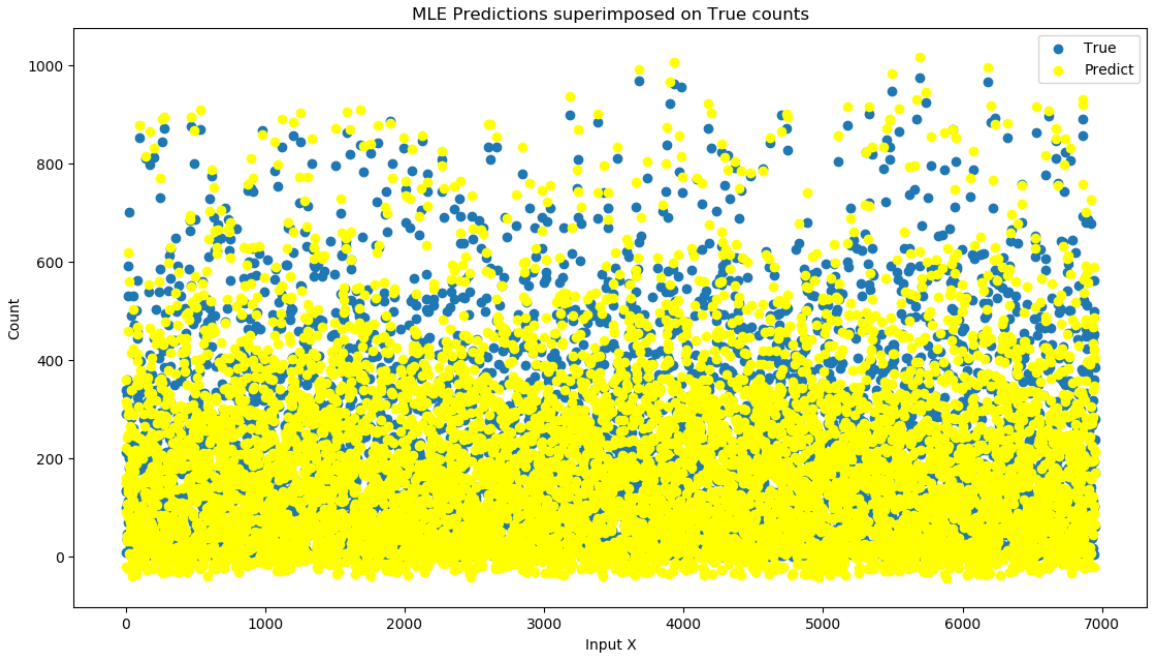


In [351]:

```
p = np.arange(0,6952,1)
fig = plt.figure(figsize = (13,7),dpi =100)
axes = fig.add_axes([1,0.1,0.8,0.8])
axes.scatter(p,t_test,label = 'True')
axes.scatter(p,y,color ='yellow',label='Predict')
axes.set_xlabel('Input X')
axes.set_ylabel('Count')
axes.set_title('MLE Predictions superimposed on True counts')
axes.legend()
```

Out[351]:

```
<matplotlib.legend.Legend at 0x1a8bbde150>
```



MAP

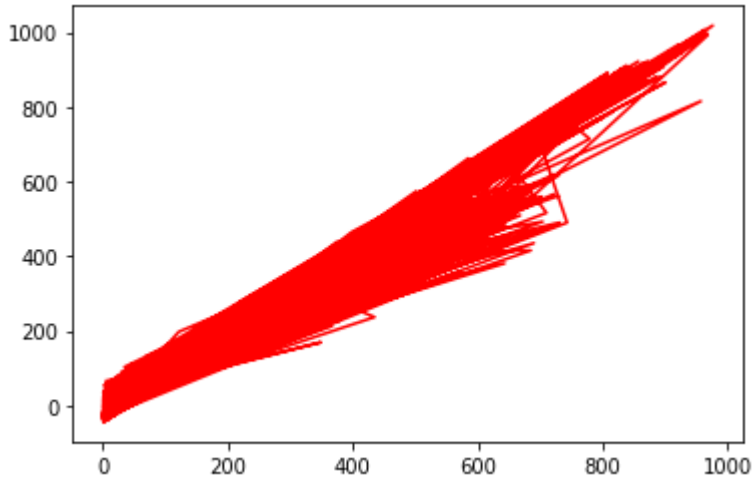
In [353]:

```
def PolynomialRegression_Regularized(l):
    X = X_train
    w = list(np.linalg.inv(X_train.T@X_train + l*np.eye(10)).T@X_train.T@t_train
)
    y = X_test@w
    print(w)
    eigvals, eigvecs = np.linalg.eig(X_test.T@X_test + l*np.eye(10))
    c = np.max(np.abs(eigvals))/np.min(np.abs(eigvals))
    return y,c
```

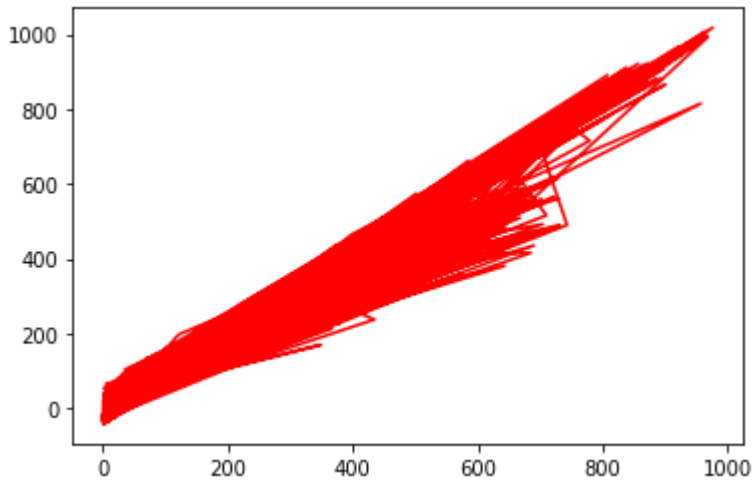
In [354]:

```
l = [0, 10,50,120,145,155,175,200,400]

for i in range(len(l)):
    y,c = PolynomialRegression_Regularized(l[i])
    print("Condition Number = "+str(c))

    plt.plot(t_test, y, 'r', label = 'Prediction')
    plt.show()
```

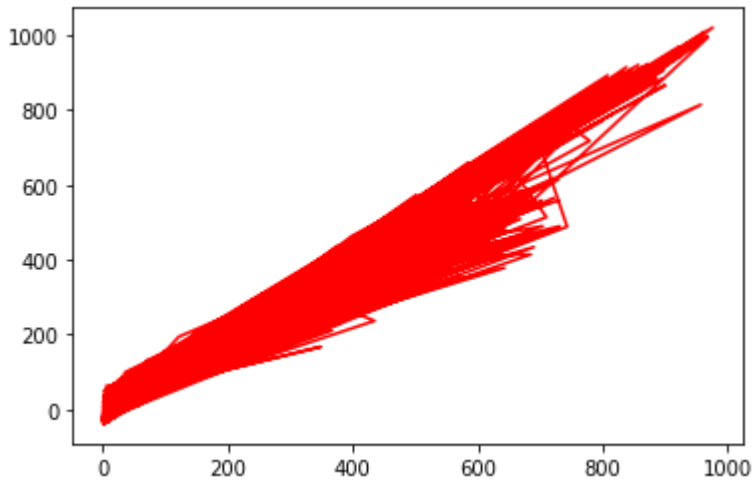
```
[3.5920038155522676, 0.09864616210590399, 0.6868847030831747, -9.480
136751071448, -37.11035515370779, 3.2361013895486126, 94.80308399578
67, -37.47015279464357, 18.55391682056033, 1.1178614750555274]
Condition Number = 2982817.3532334943
```



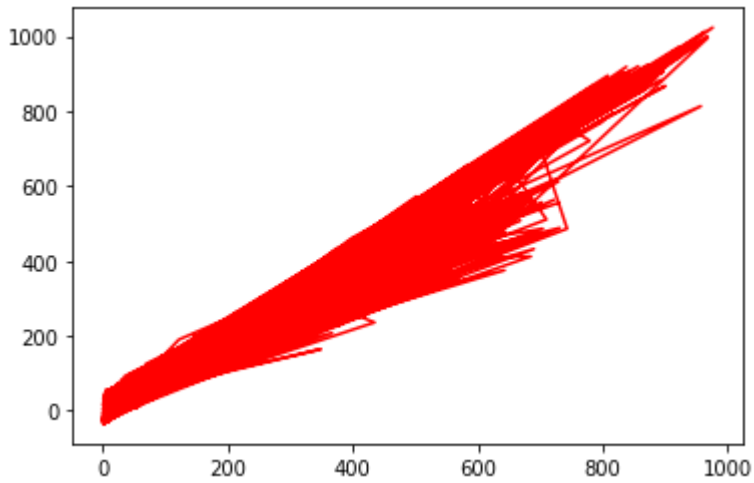

```
[3.6075763727687478, 0.11275471007097587, 0.703757496228371, -9.0367
56558093076, -36.85313580448049, 3.039699761537025, 92.1768503554920
1, -35.88611239770134, 18.005778326942064, 1.1191490772020778]
Condition Number = 2732529.389211288
```



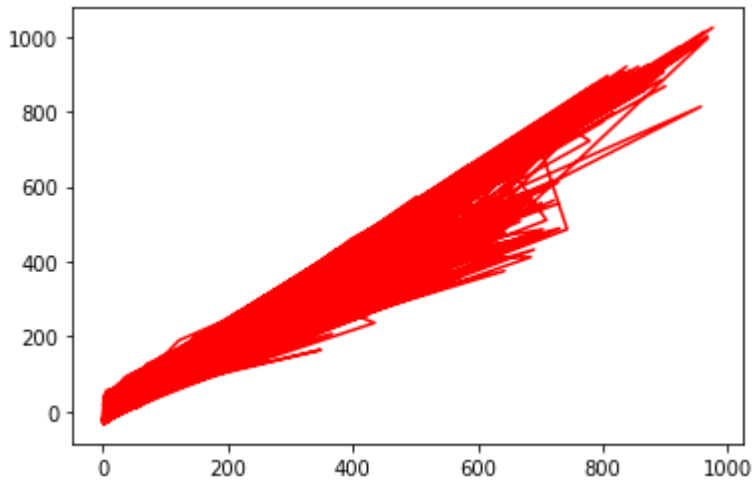

```
[3.6633870451506567, 0.16685320702408507, 0.7626484627753681, -7.532
953319686207, -35.88165860639205, 2.398516032945362 8, 83.03011910329
84, -30.58964043919171, 16.09399311063592, 1.1235583847192279]
Condition Number = 2045858.2801715925
```



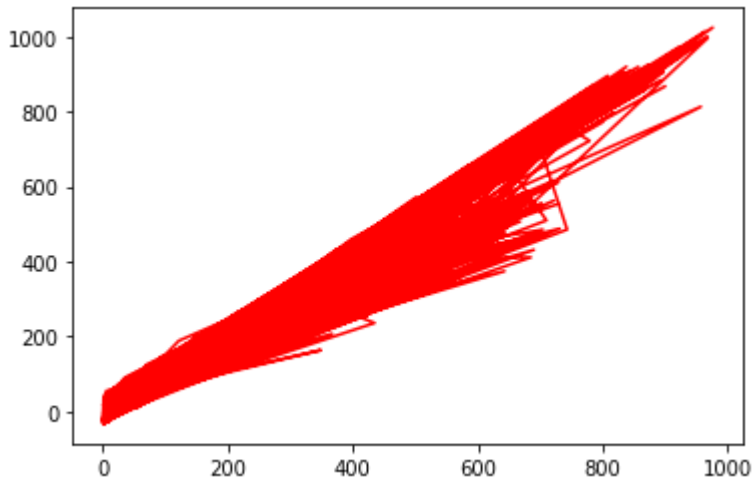

```
[3.7353229182456964, 0.2512328663905999, 0.8412318645023326, -5.6428
05501527078, -34.36748263348045, 1.64854841050981, 70.8392696206438
3, -24.097020199343675, 13.55521314441105, 1.129230434232674]
Condition Number = 1420965.6375298258
```



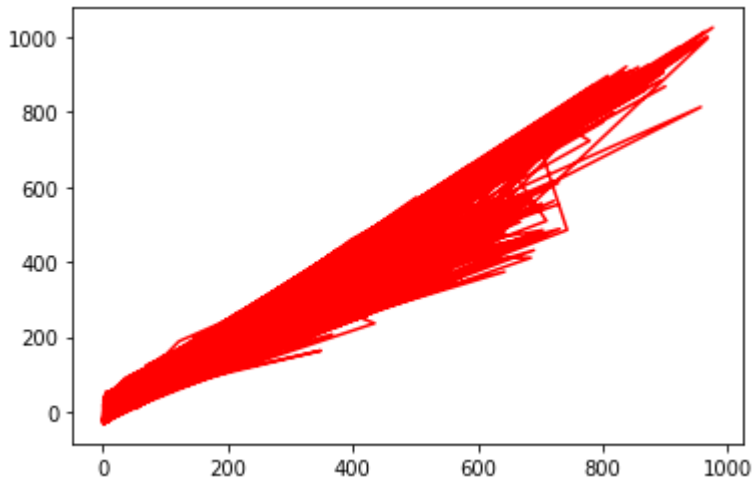

```
[3.753586047297405, 0.27816179352121084, 0.8638341029125932, -5.1261
79479485124, -33.874211774898484, 1.4544709130255944, 67.32672088909
766, -22.353766690489138, 12.828506232332035, 1.1308153512663188]
Condition Number = 1281203.4160085043
```



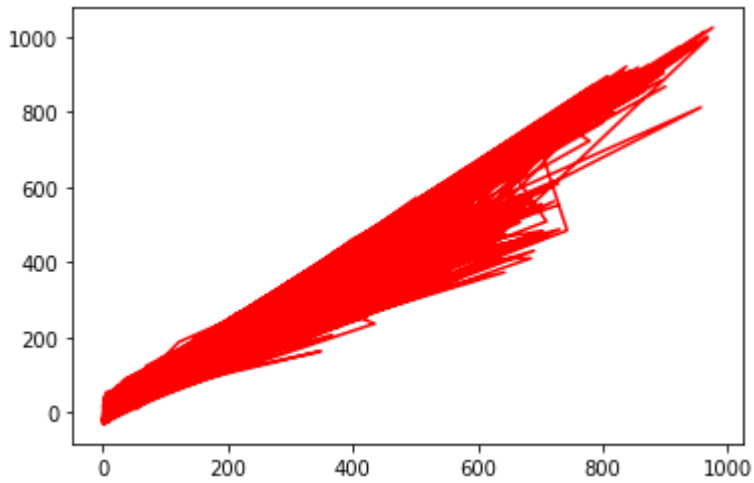

```
[3.759894101802973, 0.28848207915867863, 0.8722381223838322, -4.9373
49835602573, -33.68299776199147, 1.3846754277150057, 66.019002565844
69, -21.71987484114213, 12.558685995836221, 1.1313992181774546]
Condition Number = 1232705.2399108405
```



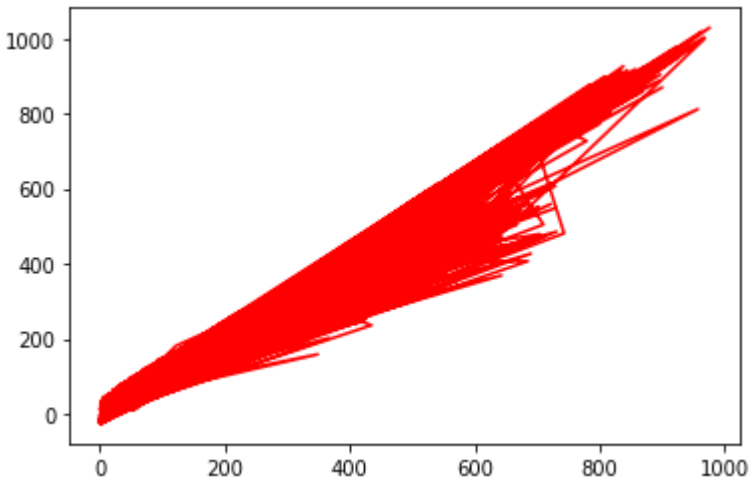

```
[3.770897007277836, 0.30837666909276645, 0.8880685735699982, -4.5865
77827510392, -33.31033876349888, 1.2565993901410573, 63.5524334230142
4, -20.546916842588978, 12.050945731712343, 1.1324909292898213]
Condition Number = 1145948.6577939538
```




```
[3.7818056218201574, 0.33190207529511045, 0.9062060755330956, -4.192
761201789534, -32.861691447232026, 1.1151556290253093, 60.7198427826
5312, -19.23691918526082, 11.469913254636332, 1.1337285868465201]
Condition Number = 1053287.0385865236
```




```
[3.783326954169282, 0.47622924511119336, 1.006656546465371, -2.17855
7435864309, -29.80917121620012, 0.4190296815824497, 44.7800279978368
1, -12.630510050755449, 8.249422915707532, 1.1403209839010695]
Condition Number = 639564.7427364199
```


1000
800
600
400
200
0
0
200
400
600
800
1000

In [356]:

```
l = [0, 10,50,120,145,150,175,200,400]
p = np.arange(0,6952,1)
for i in range(len(l)):
    y,c = PolynomialRegression_Regularized(l[i])
    print("Condition Number = "+str(c))

    e = abs(np.array(y)-np.array(t_test))
    print("Absolute_Error",e.sum(),"\nLamda_value",l[i],"\nR2_Score",r2_score(t_
test,y),"\n")
    fig = plt.figure(figsize = (13,7),dpi =100)
    axes = fig.add_axes([1,0.1,0.8,0.8])
    axes.scatter(p,t_test)
    axes.scatter(p,y,color ='yellow')
```

[3.5920038155522676, 0.09864616210590399, 0.6868847030831747, -9.480
136751071448, -37.11035515370779, 3.2361013895486126, 94.80308399578
67, -37.47015279464357, 18.55391682056033, 1.1178614750555274]
Condition Number = 2982817.3532334943
Absolute_Error 156340.74532808247
Lamda_value 0
R2_Score 0.9657434037223945

[3.6075763727687478, 0.11275471007097587, 0.703757496228371, -9.0367
56558093076, -36.85313580448049, 3.039699761537025, 92.1768503554920
1, -35.88611239770134, 18.005778326942064, 1.1191490772020778]
Condition Number = 2732529.389211288
Absolute_Error 155864.6501554687
Lamda_value 10
R2_Score 0.9657238055921746

[3.6633870451506567, 0.16685320702408507, 0.7626484627753681, -7.532
953319686207, -35.88165860639205, 2.3985160329453628, 83.03011910329
84, -30.58964043919171, 16.09399311063592, 1.1235583847192279]
Condition Number = 2045858.2801715925
Absolute_Error 154618.23000265742
Lamda_value 50
R2_Score 0.9655083023374834

[3.7353229182456964, 0.2512328663905999, 0.8412318645023326, -5.6428
05501527078, -34.36748263348045, 1.64854841050981, 70.8392696206438
3, -24.097020199343675, 13.55521314441105, 1.129230434232674]
Condition Number = 1420965.6375298258
Absolute_Error 153862.20197592577
Lamda_value 120
R2_Score 0.9648699181153494

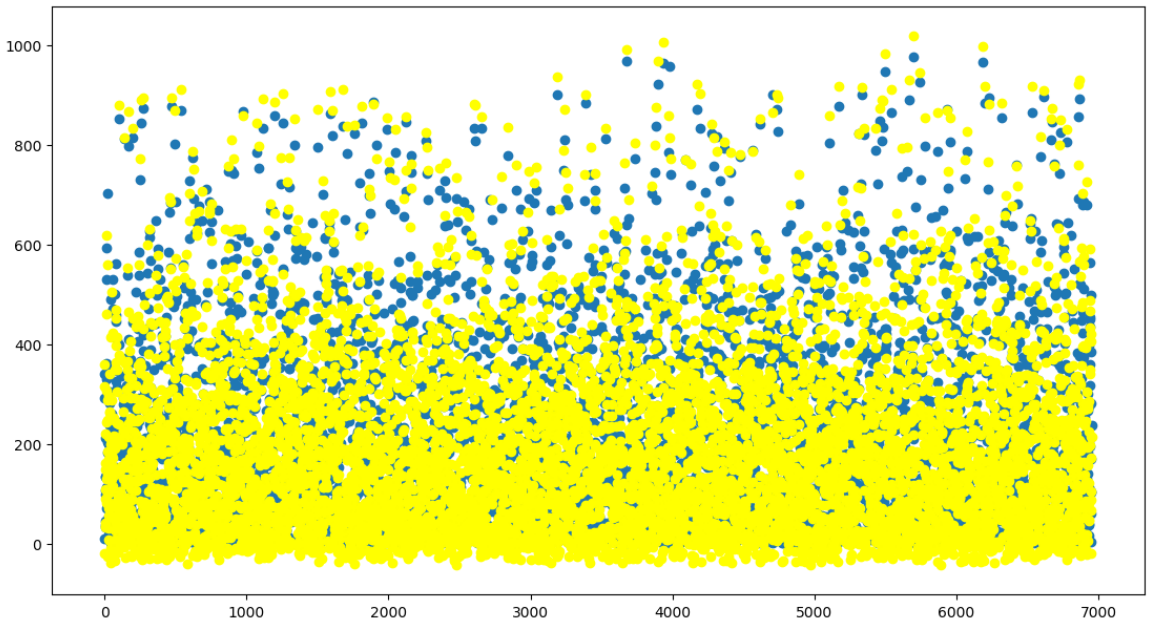
[3.753586047297405, 0.27816179352121084, 0.8638341029125932, -5.1261
79479485124, -33.874211774898484, 1.4544709130255944, 67.32672088909
766, -22.353766690489138, 12.828506232332035, 1.1308153512663188]
Condition Number = 1281203.4160085043
Absolute_Error 153877.72526619243
Lamda_value 145
R2_Score 0.9646120115393431

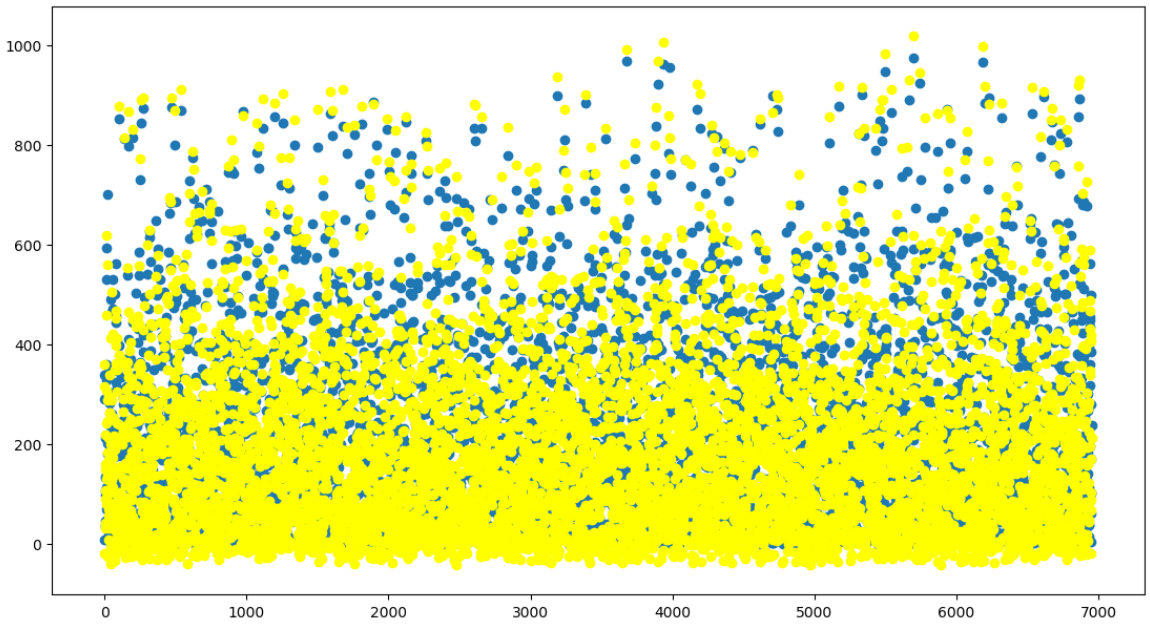
[3.7568091210654577, 0.2833535633508539, 0.868078875735625, -5.03057
9616248738, -33.77818541011512, 1.4190598550155715, 66.6663407275177
3, -22.032625569822066, 12.692198231769972, 1.1311106268004814]
Condition Number = 1256486.515624784
Absolute_Error 153888.6803268979
Lamda_value 150
R2_Score 0.9645598217788944

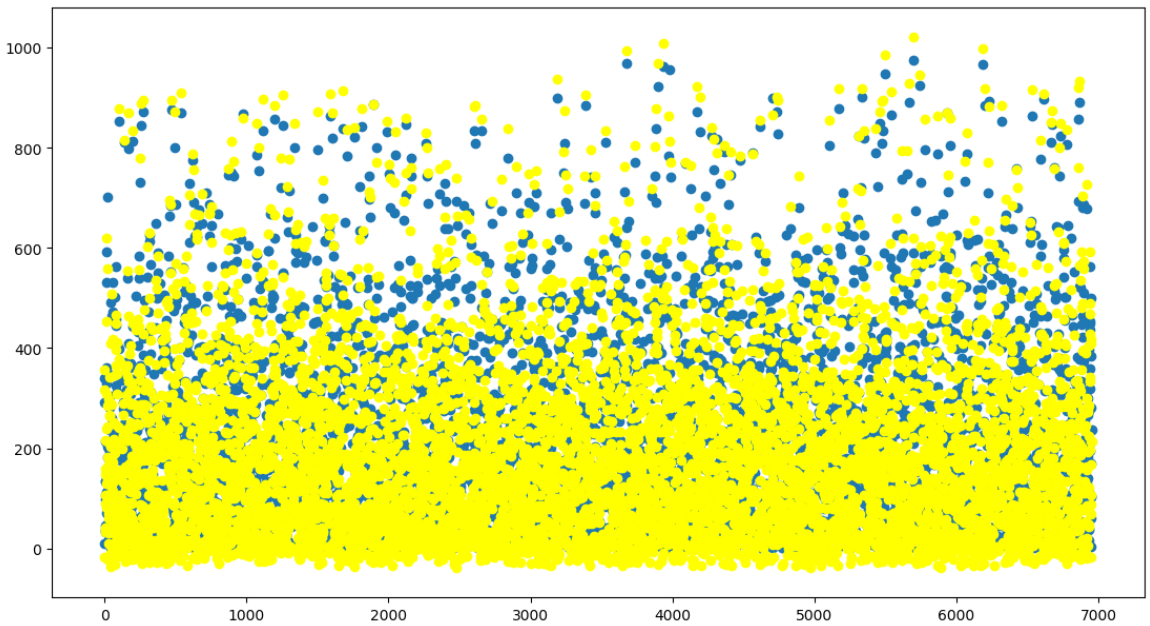
[3.770897007277836, 0.30837666909276645, 0.8880685735699982, -4.5865
77827510392, -33.3103387634988, 1.2565993901410573, 63.5524334230142
4, -20.546916842588978, 12.050945731712343, 1.1324909292898213]
Condition Number = 1145948.6577939538
Absolute_Error 153986.1990899599
Lamda_value 175
R2_Score 0.9642978829892463

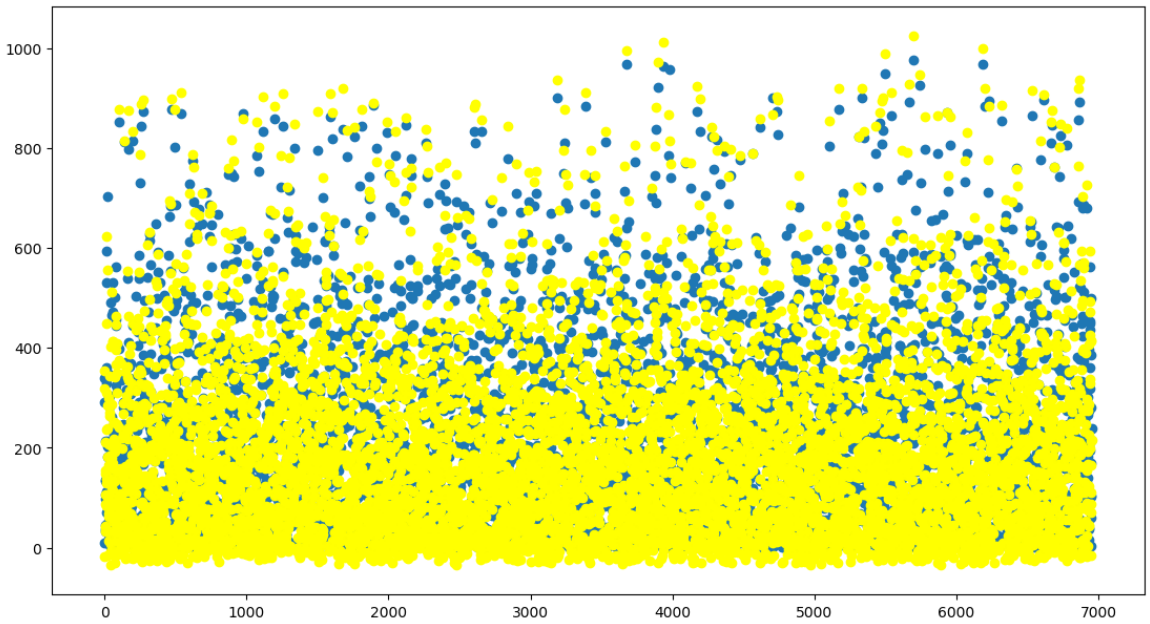
[3.7818056218201574, 0.33190207529511045, 0.9062060755330956, -4.192
761201789534, -32.861691447232026, 1.1151556290253093, 60.7198427826
5312, -19.23691918526082, 11.469913254636332, 1.1337285868465201]
Condition Number = 1053287.0385865236
Absolute_Error 154149.34154990467

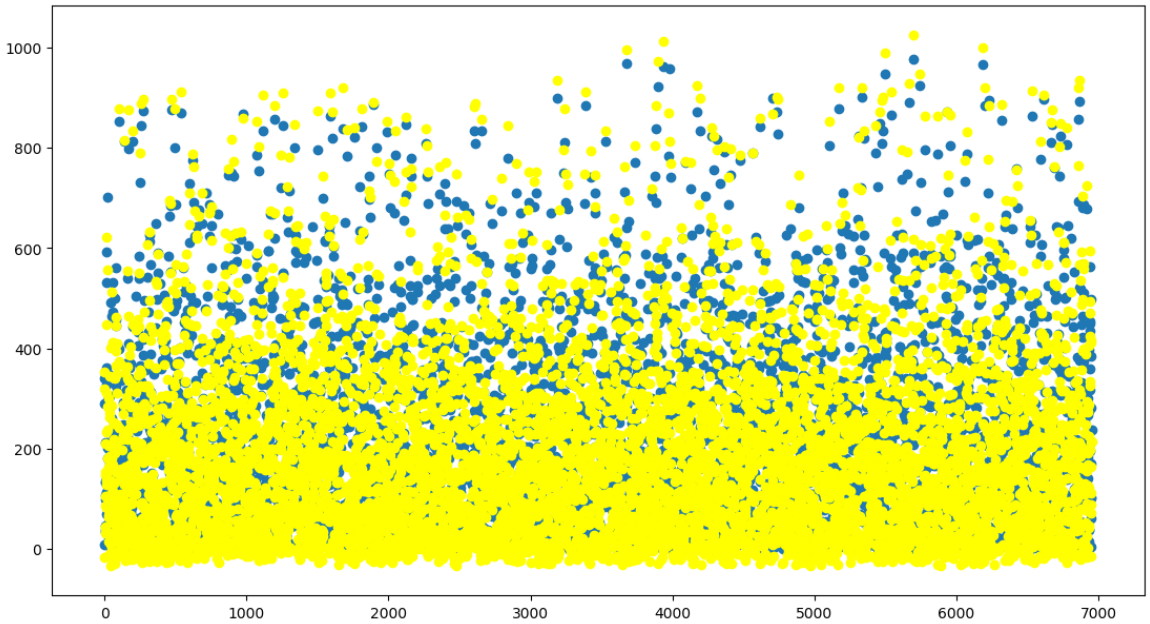
```
Lamda_value 200
R2_Score 0.9640367760203078

[3.783326954169282, 0.47622924511119336, 1.006656546465371, -2.17855
7435864309, -29.80917121620012, 0.4190296815824497, 44.7800279978368
1, -12.630510050755449, 8.249422915707532, 1.1403209839010695]
Condition Number = 639564.7427364199
Absolute_Error 156170.5297782367
Lamda_value 400
R2_Score 0.9621462699229747
```

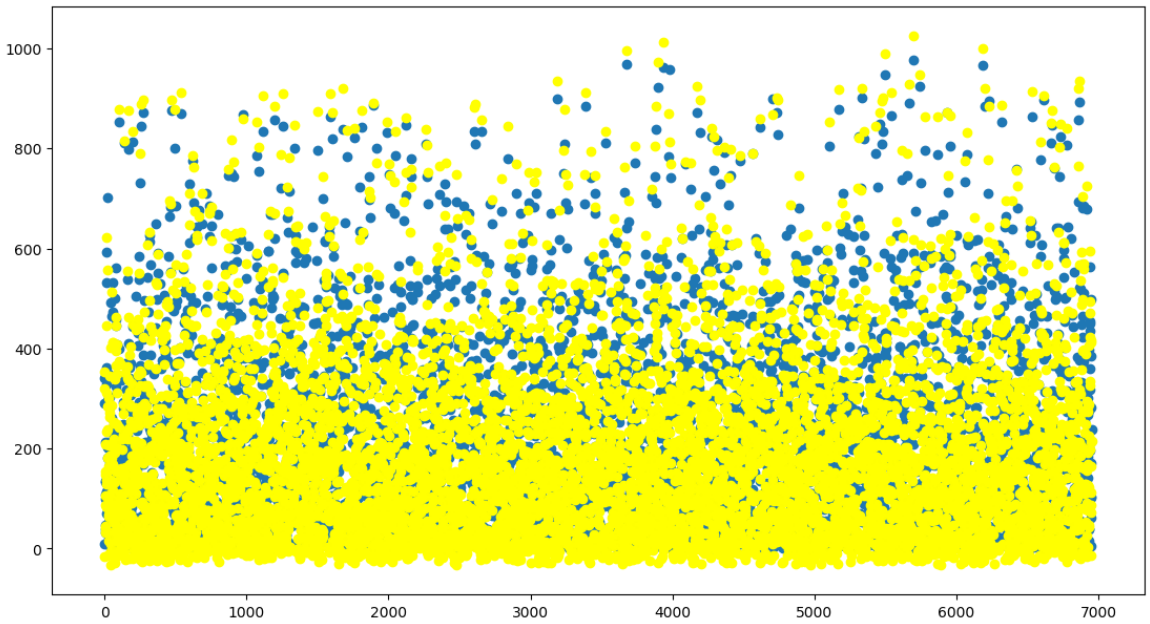


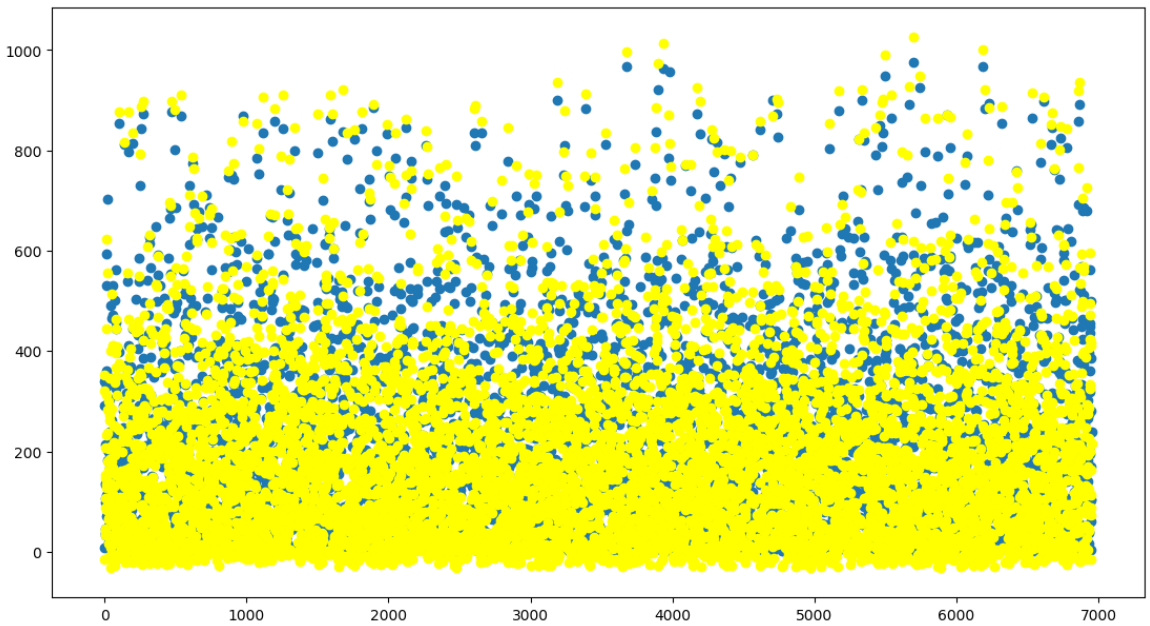

1000
800
600
400
200
0
0
1000
2000
3000
4000
5000
6000
7000

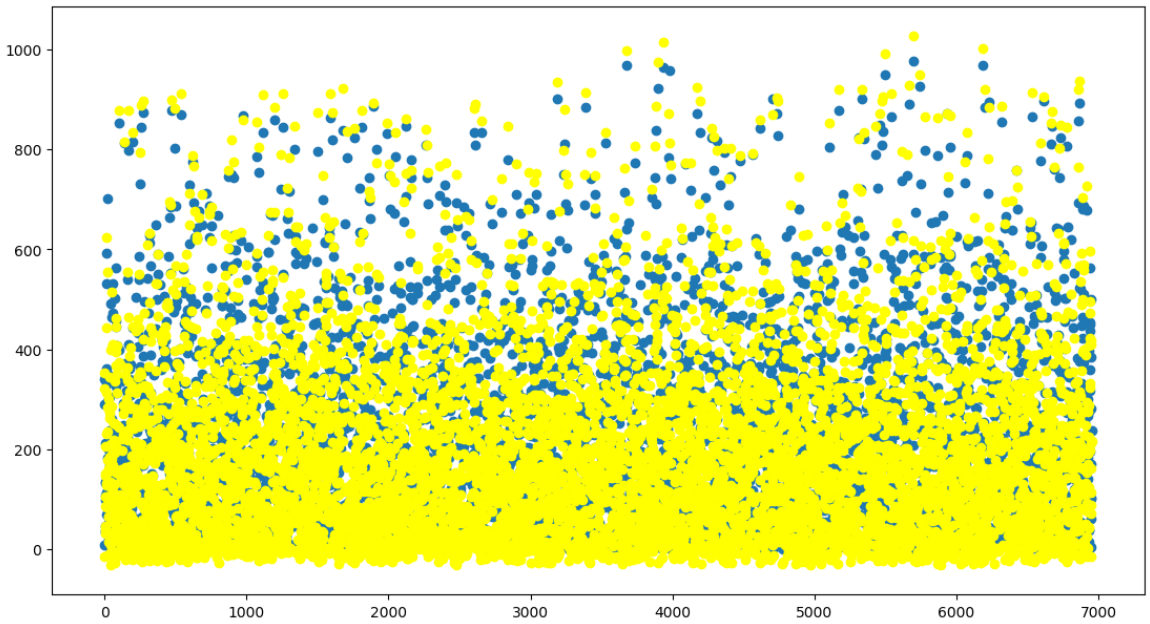

1000
800
600
400
200
0
0
1000
2000
3000
4000
5000
6000
7000

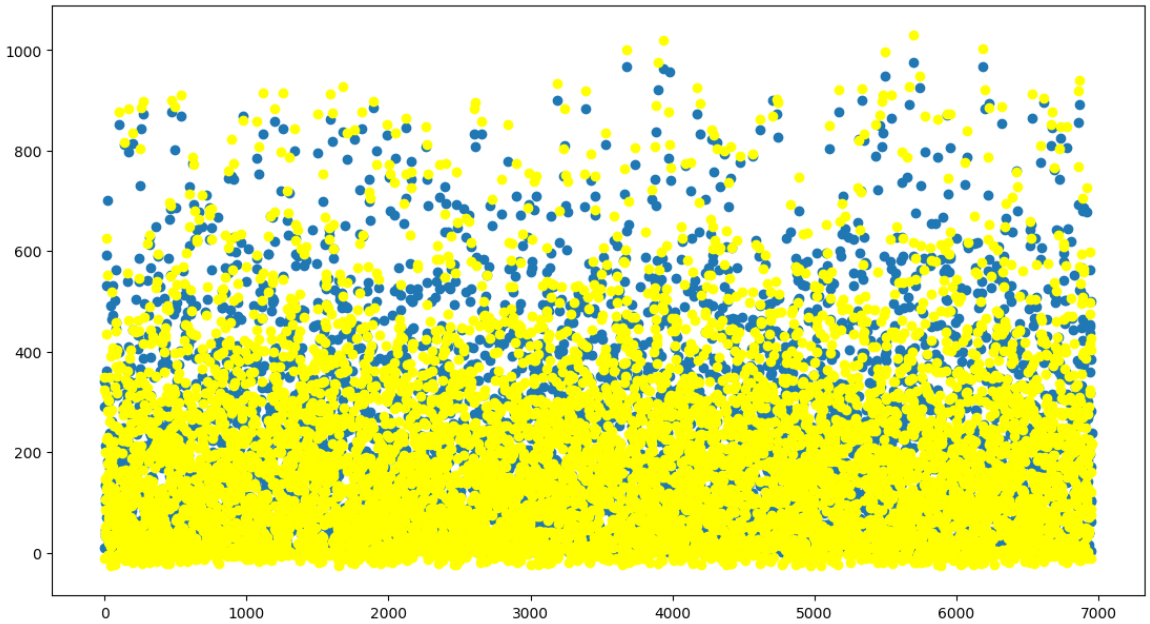

1000
800
600
400
200
0
0
1000
2000
3000
4000
5000
6000
7000


1000
800
600
400
200
0
0
1000
2000
3000
4000
5000
6000
7000


1000
800
600
400
200
0
0
1000
2000
3000
4000
5000
6000
7000


1000
800
600
400
200
0
0
1000
2000
3000
4000
5000
6000
7000


1000
800
600
400
200
0
0
1000
2000
3000
4000
5000
6000
7000


1000
800
600
400
200
0
0
1000
2000
3000
4000
5000
6000
7000

In [337]:

```
from sklearn import metrics
import pylab
import scipy.stats as stats

#Choosing LAMBDA = 120

y,c = PolynomialRegression_Regularized(120)
print("Condition Number = "+str(c))
e = abs(np.array(y)-np.array(t_test))
print("Absolute_Error",e.sum(),"\nLamda_value",l[i],"\nR2_Score",r2_score(t_test
,y),"\n")
```

```
[3.7353229182456964, 0.2512328663905999, 0.8412318645023326, -5.6428
05501527078, -34.36748263348045, 1.64854841050981, 70.8392696206438
3, -24.097020199343675, 13.55521314441105, 1.129230434232674]
Condition Number = 1420965.6375298258
Absolute_Error 153862.20197592577
Lamda_value 400
R2_Score 0.9648699181153494
```

In [338]:

```
#MEAN SQUARED ERROR

metrics.mean_squared_error(t_test,y)
```

Out[338]:

```
1155.9008955228812
```

In [339]:

```
#MEDIAN SQUARED ERROR

metrics.median_absolute_error(t_test,y)
```

Out[339]:

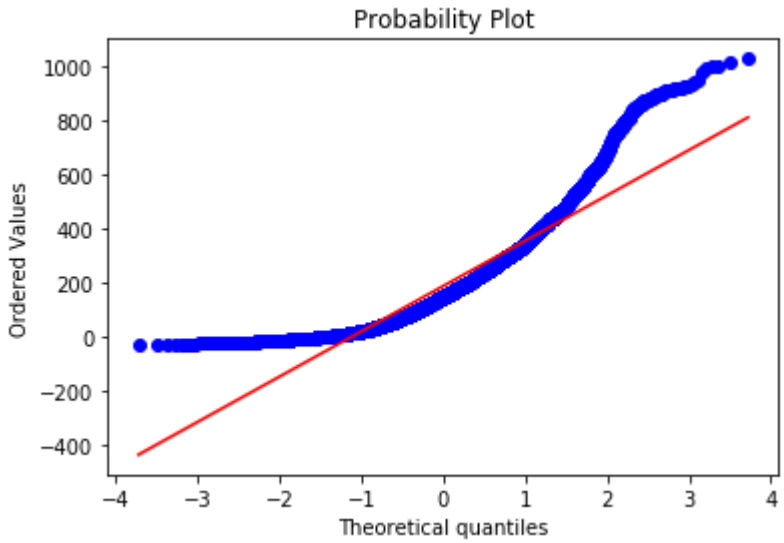
```
15.436259210259827
```

In [355]:

```
#Q-Q PLOT
stats.probplot(y,dist='norm', plot = pylab)
```

Out[355]:

```
((array([-3.71977607, -3.48945422, -3.36275798, ...,  3.36275798,
          3.48945422,  3.71977607]),
  array([ -26.35168521,  -26.2340296 ,  -25.96227394, ..., 1003.4369
1197,
         1018.80829981, 1029.16748935])),
 (167.6347922648931, 188.22348629338637, 0.9374011342865406))
```



After plotting the above values of condition number, Lamda, Error and R2-Score we can observe that the error is least for the value Lamda = 120 and then starts increasing again. Hence we Choose the MAP method over the MLE @ Lambda = 120 as the error is least for this value.

In [ ]: