HW1_Polynomial_Regression

April 8, 2020

0.1 Week1 HW: Polynomial Regression

Minju Jo

```
In [1]: import numpy as np
        np.set_printoptions(precision=3)
        import pandas as pd
        pd.set_option('display.precision',3)
        import matplotlib.pyplot as plt
        import seaborn as sns
```

```
In [2]: bike = pd.read_csv('day.csv')
        bike.head()
```

Out[2]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	6	0	2	0.344	0.364	0.806	0.160	331	654	985
1	2	2011-01-02	1	0	1	0	0	0	2	0.363	0.354	0.696	0.249	131	670	801
2	3	2011-01-03	1	0	1	0	1	1	1	0.196	0.189	0.437	0.248	120	1229	1349
3	4	2011-01-04	1	0	1	0	2	1	1	0.200	0.212	0.590	0.160	108	1454	1562
4	5	2011-01-05	1	0	1	0	3	1	1	0.227	0.229	0.437	0.187	82	1518	1600

```
In [3]: bike.shape
```

```
Out[3]: (731, 16)
```

```
In [4]: bike.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
```

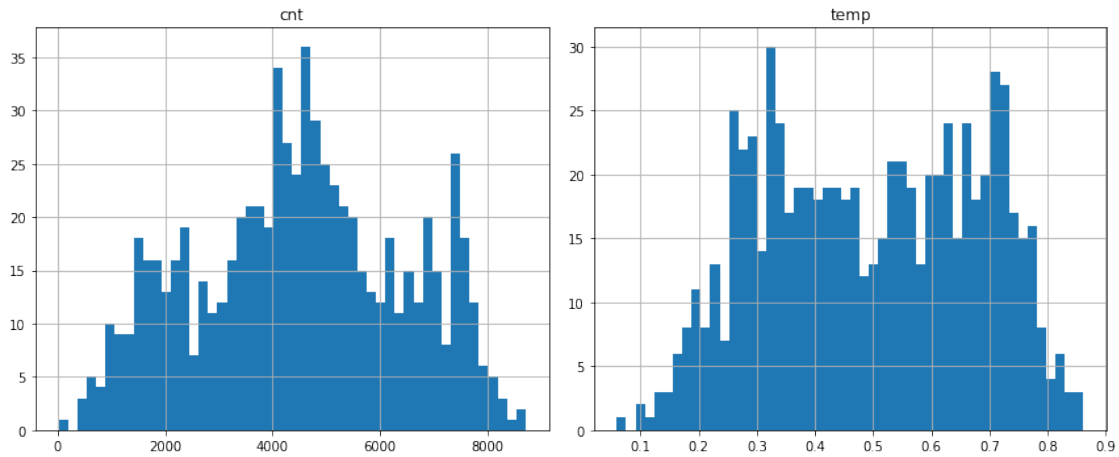
```
 0   instant     731 non-null    int64
 1   dteday      731 non-null    object
 2   season      731 non-null    int64
 3   yr          731 non-null    int64
 4   mnth        731 non-null    int64
 5   holiday     731 non-null    int64
 6   weekday     731 non-null    int64
 7   workingday  731 non-null    int64
 8   weathersit  731 non-null    int64
 9   temp        731 non-null    float64
 10  atemp       731 non-null    float64
 11  hum         731 non-null    float64
 12  windspeed   731 non-null    float64
 13  casual      731 non-null    int64
 14  registered  731 non-null    int64
 15  cnt         731 non-null    int64
dtypes: float64(4), int64(11), object(1)
memory usage: 91.5+ KB
```

```
In [5]: data= bike[['cnt','temp']]
        data.describe()
```

Out[5]:

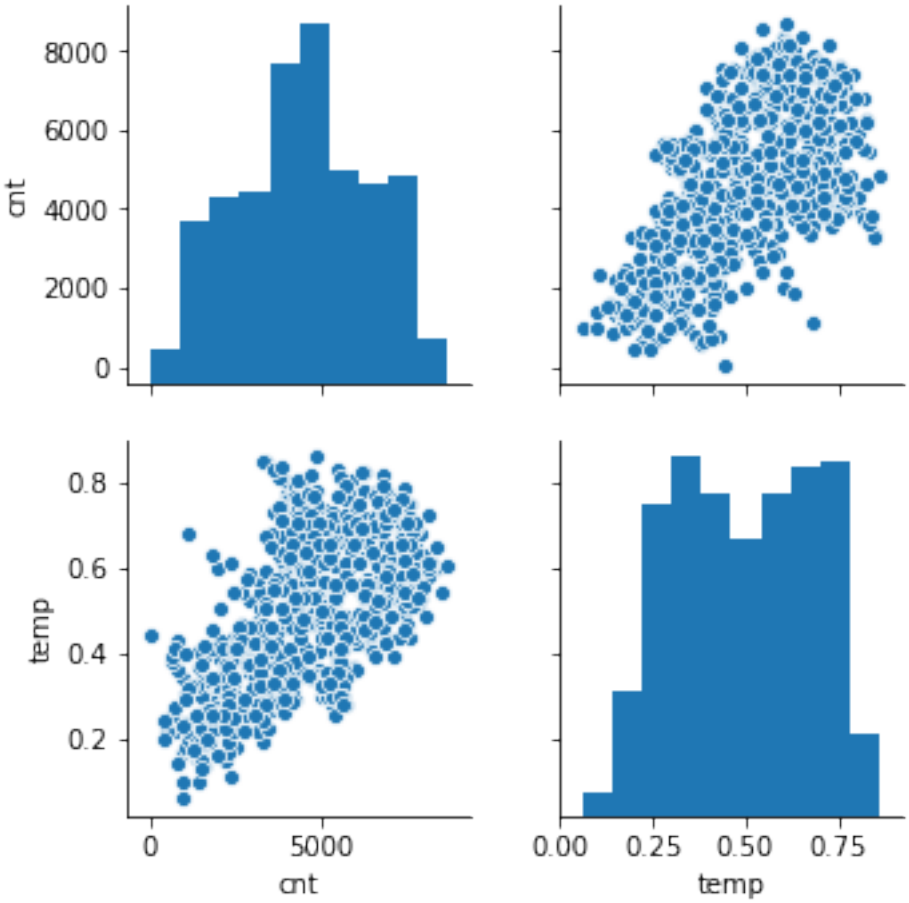
	cnt	temp
count	731.000	731.000
mean	4504.349	0.495
std	1937.211	0.183
min	22.000	0.059
25%	3152.000	0.337
50%	4548.000	0.498
75%	5956.000	0.655
max	8714.000	0.862

```
In [6]: data.hist(bins=50, figsize=(12,5))
        plt.tight_layout()
```



```
In [7]: sns.pairplot(data)
```

```
Out[7]: <seaborn.axisgrid.PairGrid at 0x183ce876f28>
```



0.1.1 Polynomial Regression

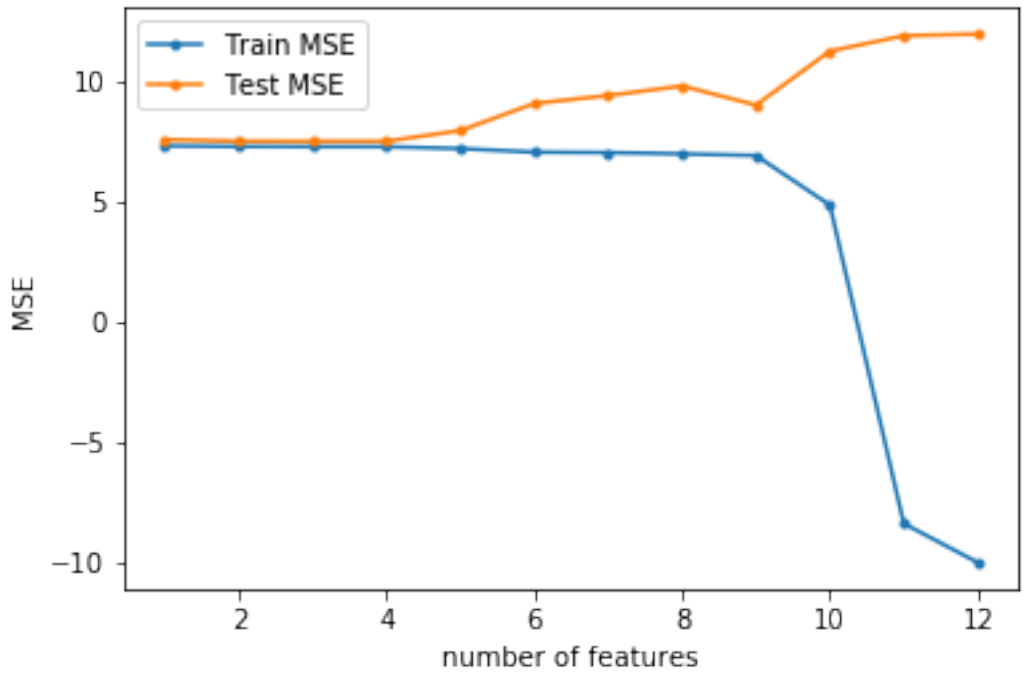
```
In [8]: d_bike = data.sample(n=30).reset_index()
```

```
In [9]: X=np.hstack([np.ones((30,1)), d_bike[["temp"]].to_numpy()])
        Y=d_bike["cnt"]
```

```
In [10]: def mse(actual_y, predicted_y):
             return np.log(np.sqrt(np.mean((actual_y-predicted_y)**2)))
```

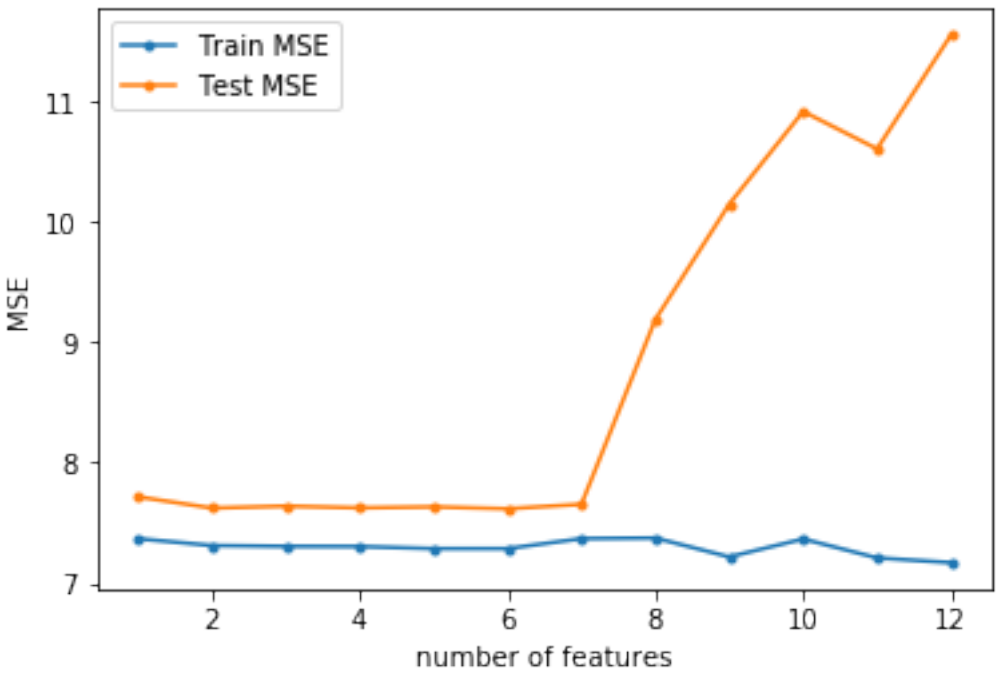
```
In [11]: from sklearn.model_selection import train_test_split
         X_train,X_test = train_test_split(X,test_size=0.6,random_state=47)
         Y_train,Y_test = train_test_split(Y,test_size=0.6,random_state=47)
```

```
In [12]: import sklearn.linear_model as lm
         from sklearn.preprocessing import PolynomialFeatures
         linear_model = lm.LinearRegression(fit_intercept=False)

In [13]: tr_errors = []
         te_errors = []

         for N in range(1, 13):
             poly = PolynomialFeatures(degree=N)
             X_train_poly = poly.fit_transform(X_train)
             X_test_poly = poly.fit_transform(X_test)

             linear_model.fit(X_train_poly, Y_train)

             train_error = mse(Y_train, linear_model.predict(X_train_poly))
             tr_errors.append(train_error)

             test_error = mse(Y_test, linear_model.predict(X_test_poly))
             te_errors.append(test_error)

In [14]: plt.plot(range(1, 13), tr_errors, marker='.')
         plt.plot(range(1, 13), te_errors, marker='.')
         plt.legend(["Train MSE", "Test MSE"])
         plt.xlabel("number of features")
         plt.ylabel("MSE");
```



if test_size=0.2

```
In [16]: X_train,X_test = train_test_split(X,test_size=0.2,random_state=47)
         Y_train,Y_test = train_test_split(Y,test_size=0.2,random_state=47)

         tr_errors = []
         te_errors = []

         for N in range(1, 13):
             poly = PolynomialFeatures(degree=N)
             X_train_poly = poly.fit_transform(X_train)
             X_test_poly = poly.fit_transform(X_test)

             linear_model.fit(X_train_poly, Y_train)

             train_error = mse(Y_train, linear_model.predict(X_train_poly))
             tr_errors.append(train_error)

             test_error = mse(Y_test, linear_model.predict(X_test_poly))
             te_errors.append(test_error)

         plt.plot(range(1, 13), tr_errors, marker='.')
         plt.plot(range(1, 13), te_errors, marker='.')
         plt.legend(["Train MSE", "Test MSE"])
         plt.xlabel("number of features")
         plt.ylabel("MSE");
```



In [ ]: