Week 4 - Regression Analysis - Python-HR_Data

October 3, 2018

1 Data Warehousing and Data Mining

1.1 Labs

1.1.1 Prepared by Gilroy Gordon

Contact Information SCIT ext. 3643
ggordonutech@gmail.com
gilroy.gordon@utech.edu.jm

1.1.2 Week 3 - Regression Analysis in Python

Additional Reference Resources:
http://scikit-learn.org/stable/modules/linear_model.html

1.2 Objectives

```
> Data Transformation
> Data Mining
     > Linear Regression
> Model Evaluation and Prediction
     > Train/Test Split - 70/30
> Presentation
     > Scatter Plot
```

1.3 Import required libraries and acquire data

NB. The data required was retrieved from the required text for this course. This should assist you in following the concepts from the book better

```
In [1]: # import required libraries
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline
```

In [2]:
```
data_path = './data/hr_data.csv' # Path to data file
data = pd.read_csv(data_path)
data.head(15)
```

Out[2]:
```
    satisfaction_level  last_evaluation  number_project  average_montly_hours  \
0                 0.38             0.53               2                   157
1                 0.80             0.86               5                   262
2                 0.11             0.88               7                   272
3                 0.72             0.87               5                   223
4                 0.37             0.52               2                   159
5                 0.41             0.50               2                   153
6                 0.10             0.77               6                   247
7                 0.92             0.85               5                   259
8                 0.89             1.00               5                   224
9                 0.42             0.53               2                   142
10                0.45             0.54               2                   135
11                0.11             0.81               6                   305
12                0.84             0.92               4                   234
13                0.41             0.55               2                   148
14                0.36             0.56               2                   137

    time_spend_company  Work_accident  left  promotion_last_5years  sales  \
0                    3              0     1                      0  sales
1                    6              0     1                      0  sales
2                    4              0     1                      0  sales
3                    5              0     1                      0  sales
4                    3              0     1                      0  sales
5                    3              0     1                      0  sales
6                    4              0     1                      0  sales
7                    5              0     1                      0  sales
8                    5              0     1                      0  sales
9                    3              0     1                      0  sales
10                   3              0     1                      0  sales
11                   4              0     1                      0  sales
12                   5              0     1                      0  sales
13                   3              0     1                      0  sales
14                   3              0     1                      0  sales

    salary
0      low
1   medium
2   medium
3      low
4      low
5      low
6      low
7      low
8      low
```

```
9       low
10      low
11      low
12      low
13      low
14      low
```

In [3]:
```
# What columns are in the data set ? Do they have spaces that I should consider
data.columns
```

Out[3]:
```
Index(['satisfaction_level', 'last_evaluation', 'number_project',
       'average_montly_hours', 'time_spend_company', 'Work_accident', 'left',
       'promotion_last_5years', 'sales', 'salary'],
      dtype='object')
```

1.4 Aim: Can we determine a person's `Satisfaction Level` based on the other factors?

age = a(last_evaluation) + b(number_project) + c(average_montly_hours) + d(time_spend_company)

The coefficients a-d, what are they? What is the relationship between the variables? Does multicolinearity exist?

I have created a function below `create_label_encoder_dict` to assist with this. The function accepts a dataframe object and uses the `LabelEncoder` class from `sklearn.preprocessing` to encode (dummy encoding) or transform non-numerical columns to numbers. Finally it returns a dictionary object of all the encoders created for each column.

The LabelEncoder is a useful resource as it not only automatically transforms all values in a column but also keeps a track of what values were transformed from. i.e. It will change all `Female` to `0` and all `Male` to `1`

In [4]:
```
def create_label_encoder_dict(df):
    from sklearn.preprocessing import LabelEncoder

    label_encoder_dict = {}
    for column in df.columns:
        # Only create encoder for categorical data types
        if not np.issubdtype(df[column].dtype, np.number) and column != 'Age':
            label_encoder_dict[column]= LabelEncoder().fit(df[column])
    return label_encoder_dict
```

In [5]:
```
label_encoders = create_label_encoder_dict(data)
print("Encoded Values for each Label")
print("="*32)
for column in label_encoders:
    print("="*32)
    print('Encoder(%s) = %s' % (column, label_encoders[column].classes_ ))
    print(pd.DataFrame([range(0,len(label_encoders[column].classes_))], columns=label_en
```

```
Encoded Values for each Label
================================
================================
Encoder(sales) = ['IT' 'RandD' 'accounting' 'hr' 'management' 'marketing' 'product_mng'
 'sales' 'support' 'technical']
             Encoded Values
IT                        0
RandD                     1
accounting                2
hr                        3
management                4
marketing                 5
product_mng               6
sales                     7
support                   8
technical                 9
================================
Encoder(salary) = ['high' 'low' 'medium']
        Encoded Values
high                 0
low                  1
medium               2
```

In [6]:
```python
# Apply each encoder to the data set to obtain transformed values
data2 = data.copy() # create copy of initial data set
for column in data2.columns:
    if column in label_encoders:
        data2[column] = label_encoders[column].transform(data2[column])

print("Transformed data set")
print("="*32)
data2
```

```
Transformed data set
================================
```

Out[6]:
```
       satisfaction_level  last_evaluation  number_project  \
0                    0.38             0.53               2
1                    0.80             0.86               5
2                    0.11             0.88               7
3                    0.72             0.87               5
4                    0.37             0.52               2
5                    0.41             0.50               2
6                    0.10             0.77               6
7                    0.92             0.85               5
8                    0.89             1.00               5
```

9	0.42	0.53	2
10	0.45	0.54	2
11	0.11	0.81	6
12	0.84	0.92	4
13	0.41	0.55	2
14	0.36	0.56	2
15	0.38	0.54	2
16	0.45	0.47	2
17	0.78	0.99	4
18	0.45	0.51	2
19	0.76	0.89	5
20	0.11	0.83	6
21	0.38	0.55	2
22	0.09	0.95	6
23	0.46	0.57	2
24	0.40	0.53	2
25	0.89	0.92	5
26	0.82	0.87	4
27	0.40	0.49	2
28	0.41	0.46	2
29	0.38	0.50	2
...	...	...	...
14969	0.43	0.46	2
14970	0.78	0.93	4
14971	0.39	0.45	2
14972	0.11	0.97	6
14973	0.36	0.52	2
14974	0.36	0.54	2
14975	0.10	0.79	7
14976	0.40	0.47	2
14977	0.81	0.85	4
14978	0.40	0.47	2
14979	0.09	0.93	6
14980	0.76	0.89	5
14981	0.73	0.93	5
14982	0.38	0.49	2
14983	0.72	0.84	5
14984	0.40	0.56	2
14985	0.91	0.99	5
14986	0.85	0.85	4
14987	0.90	0.70	5
14988	0.46	0.55	2
14989	0.43	0.57	2
14990	0.89	0.88	5
14991	0.09	0.81	6
14992	0.40	0.48	2
14993	0.76	0.83	6
14994	0.40	0.57	2

```
14995              0.37             0.48              2
14996              0.37             0.53              2
14997              0.11             0.96              6
14998              0.37             0.52              2

       average_montly_hours  time_spend_company  Work_accident  left  \
0                       157                   3              0     1
1                       262                   6              0     1
2                       272                   4              0     1
3                       223                   5              0     1
4                       159                   3              0     1
5                       153                   3              0     1
6                       247                   4              0     1
7                       259                   5              0     1
8                       224                   5              0     1
9                       142                   3              0     1
10                      135                   3              0     1
11                      305                   4              0     1
12                      234                   5              0     1
13                      148                   3              0     1
14                      137                   3              0     1
15                      143                   3              0     1
16                      160                   3              0     1
17                      255                   6              0     1
18                      160                   3              1     1
19                      262                   5              0     1
20                      282                   4              0     1
21                      147                   3              0     1
22                      304                   4              0     1
23                      139                   3              0     1
24                      158                   3              0     1
25                      242                   5              0     1
26                      239                   5              0     1
27                      135                   3              0     1
28                      128                   3              0     1
29                      132                   3              0     1
...                     ...                 ...            ...   ...
14969                   157                   3              0     1
14970                   225                   5              0     1
14971                   140                   3              0     1
14972                   310                   4              0     1
14973                   143                   3              0     1
14974                   153                   3              0     1
14975                   310                   4              0     1
14976                   136                   3              0     1
14977                   251                   6              0     1
14978                   144                   3              0     1
14979                   296                   4              0     1
```

14980	238	5	0	1
14981	162	4	0	1
14982	137	3	0	1
14983	257	5	0	1
14984	148	3	0	1
14985	254	5	0	1
14986	247	6	0	1
14987	206	4	0	1
14988	145	3	0	1
14989	159	3	1	1
14990	228	5	1	1
14991	257	4	0	1
14992	155	3	0	1
14993	293	6	0	1
14994	151	3	0	1
14995	160	3	0	1
14996	143	3	0	1
14997	280	4	0	1
14998	158	3	0	1

	promotion_last_5years	sales	salary
0	0	7	1
1	0	7	2
2	0	7	2
3	0	7	1
4	0	7	1
5	0	7	1
6	0	7	1
7	0	7	1
8	0	7	1
9	0	7	1
10	0	7	1
11	0	7	1
12	0	7	1
13	0	7	1
14	0	7	1
15	0	7	1
16	0	7	1
17	0	7	1
18	1	7	1
19	0	7	1
20	0	7	1
21	0	7	1
22	0	7	1
23	0	7	1
24	0	7	1
25	0	7	1
26	0	7	1

```
27                          0     7     1
28                          0     2     1
29                          0     2     1
...                       ...   ...   ...
14969                       0     7     2
14970                       0     7     2
14971                       0     7     2
14972                       0     2     2
14973                       0     2     2
14974                       0     2     2
14975                       0     3     2
14976                       0     3     2
14977                       0     3     2
14978                       0     3     2
14979                       0     9     2
14980                       0     9     0
14981                       0     9     1
14982                       0     9     2
14983                       0     9     2
14984                       0     9     2
14985                       0     9     2
14986                       0     9     1
14987                       0     9     1
14988                       0     9     1
14989                       0     9     1
14990                       0     8     1
14991                       0     8     1
14992                       0     8     1
14993                       0     8     1
14994                       0     8     1
14995                       0     8     1
14996                       0     8     1
14997                       0     8     1
14998                       0     8     1

[14999 rows x 10 columns]
```

In [7]:
```
# separate our data into dependent (Y) and independent(X) variables
X_data = data2[['last_evaluation','number_project','average_montly_hours','time_spend_co
Y_data = data2['satisfaction_level']
```

1.5 70/30 Train Test Split

We will split the data using a 70/30 split. i.e. 70% of the data will be randomly chosen to train the model and 30% will be used to evaluate the model

In [8]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_data, Y_data, test_size=0.30)
```

In [9]:
```
# Import linear model package (has several regression classes)
from sklearn import linear_model
```

In [10]:
```
# Create an instance of linear regression
reg = linear_model.LinearRegression()
```

In [11]:
```
reg.fit(X_train,y_train)
```

Out[11]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [12]:
```
reg.coef_
```

Out[12]:
```
array([ 2.52091853e-01, -3.86756827e-02,  1.02855525e-04, -1.40672606e-02])
```

In [13]:
```
X_train.columns
```

Out[13]:
```
Index(['last_evaluation', 'number_project', 'average_montly_hours',
       'time_spend_company'],
      dtype='object')
```

In [14]:
```
print("Regression Coefficients")
pd.DataFrame(reg.coef_,index=X_train.columns,columns=["Coefficient"])
```

```
Regression Coefficients
```

Out[14]:

	Coefficient
last_evaluation	0.252092
number_project	-0.038676
average_montly_hours	0.000103
time_spend_company	-0.014067

In [15]:
```
# Intercept
reg.intercept_
```

Out[15]:
```
0.6078482685006579
```

In [16]:
```
# Make predictions using the testing set
test_predicted = reg.predict(X_test)
test_predicted
```

Out[16]:
```
array([0.60151576, 0.64939671, 0.52336411, ..., 0.67138193, 0.69110759,
       0.5978772 ])
```

In [17]:
```
data3 = X_test.copy()
data3['predicted_satisfaction_level']=test_predicted
data3['satisfaction_level']=y_test
data3.head()
```

Out[17]:
```
       last_evaluation  number_project  average_montly_hours  \
14667             0.93               5                   223
13373             0.89               4                   137
9246              0.55               5                   121
12261             0.97               5                   263
9359              0.54               3                   271

       time_spend_company  predicted_satisfaction_level  satisfaction_level
14667                   5                      0.601516                0.86
13373                   3                      0.649397                0.78
9246                    3                      0.523364                0.27
12261                   5                      0.615714                0.82
9359                    3                      0.613623                0.97
```

In [18]:
```
from sklearn.metrics import mean_squared_error, r2_score
```

In [19]:
```
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(y_test, test_predicted))
```

```
Mean squared error: 0.06
```

In [20]:
```
# Explained variance score: 1 is perfect prediction
# R squared
print('Variance score: %.2f' % r2_score(y_test, test_predicted))
```

```
Variance score: 0.06
```

In [21]:
```
help(reg.score)
```

```
Help on method score in module sklearn.base:

score(X, y, sample_weight=None) method of sklearn.linear_model.base.LinearRegression instance
    Returns the coefficient of determination R^2 of the prediction.

    The coefficient R^2 is defined as (1 - u/v), where u is the residual
    sum of squares ((y_true - y_pred) ** 2).sum() and v is the total
    sum of squares ((y_true - y_true.mean()) ** 2).sum().
    The best possible score is 1.0 and it can be negative (because the
    model can be arbitrarily worse). A constant model that always
    predicts the expected value of y, disregarding the input features,
    would get a R^2 score of 0.0.

    Parameters
    ----------
    X : array-like, shape = (n_samples, n_features)
        Test samples.
```

```
    y : array-like, shape = (n_samples) or (n_samples, n_outputs)
        True values for X.

    sample_weight : array-like, shape = [n_samples], optional
        Sample weights.

    Returns
    -------
    score : float
        R^2 of self.predict(X) wrt. y.
```

```
In [22]: reg.score(X_test,y_test)
```

```
Out[22]: 0.061684941263884914
```

1.5.1 Visualizations

It's difficult to plat a scatter plot with so many dimensions
How about Dimensionality Reduction?
One such method - Principal Component Analysis

```
In [23]: from sklearn.decomposition import PCA
```

```
In [24]: pca = PCA(n_components=1)
```

```
In [25]: pca.fit(data2[X_train.columns])
```

```
Out[25]: PCA(copy=True, iterated_power='auto', n_components=1, random_state=None,
           svd_solver='auto', tol=0.0, whiten=False)
```

```
In [26]: pca.components_
```

```
Out[26]: array([[0.00116455, 0.01030169, 0.99993927, 0.00373904]])
```

```
In [27]: pca.n_features_
```

```
Out[27]: 4
```

```
In [28]: pca.n_components_
```

```
Out[28]: 1
```

Now that we can reduce our components(factors/features) let us plot (X against y)

```
In [29]: #Again :
         X_test
```

Out[29]:

	last_evaluation	number_project	average_montly_hours	\
14667	0.93	5	223	
13373	0.89	4	137	
9246	0.55	5	121	
12261	0.97	5	263	
9359	0.54	3	271	
2419	0.66	6	164	
11647	0.63	4	104	
13380	0.83	5	216	
824	0.56	2	138	
14026	0.81	4	179	
9008	0.54	3	159	
12126	0.49	2	132	
811	0.97	7	288	
6432	0.67	2	136	
5145	0.92	2	198	
9546	0.95	4	137	
12330	0.57	2	140	
3371	0.52	5	222	
181	0.84	6	261	
9228	0.49	3	267	
6520	0.70	4	221	
5076	0.71	5	222	
6163	0.84	3	239	
7981	0.89	4	255	
7128	0.99	5	208	
2819	0.37	2	159	
13084	0.71	4	268	
4388	0.96	4	143	
9197	0.65	3	183	
1291	0.90	6	272	
...	...	...	...	
9210	0.88	5	223	
343	0.56	2	143	
2031	0.57	2	160	
12730	0.47	2	128	
7305	0.67	3	113	
2380	0.60	3	205	
8912	0.53	4	181	
1851	0.53	2	147	
9529	0.88	5	225	
4124	0.97	3	199	
11594	0.55	3	271	
12935	0.62	3	152	
14351	0.96	6	245	
3684	0.56	4	214	
14646	0.47	2	135	
8283	0.96	4	287	

12430	0.65	5	195
7600	0.99	4	152
10151	0.73	3	195
4979	0.36	4	97
7516	0.97	4	264
1167	0.50	2	127
11364	0.77	3	144
12793	0.50	4	156
12464	0.98	5	234
8714	0.82	4	190
6409	0.75	4	263
9903	0.87	4	263
4778	0.90	3	142
11310	0.74	5	243

	time_spend_company
14667	5
13373	3
9246	3
12261	5
9359	3
2419	5
11647	7
13380	4
824	3
14026	3
9008	3
12126	3
811	4
6432	6
5145	2
9546	4
12330	3
3371	2
181	4
9228	3
6520	5
5076	3
6163	3
7981	3
7128	2
2819	6
13084	3
4388	3
9197	3
1291	5
...	...
9210	3

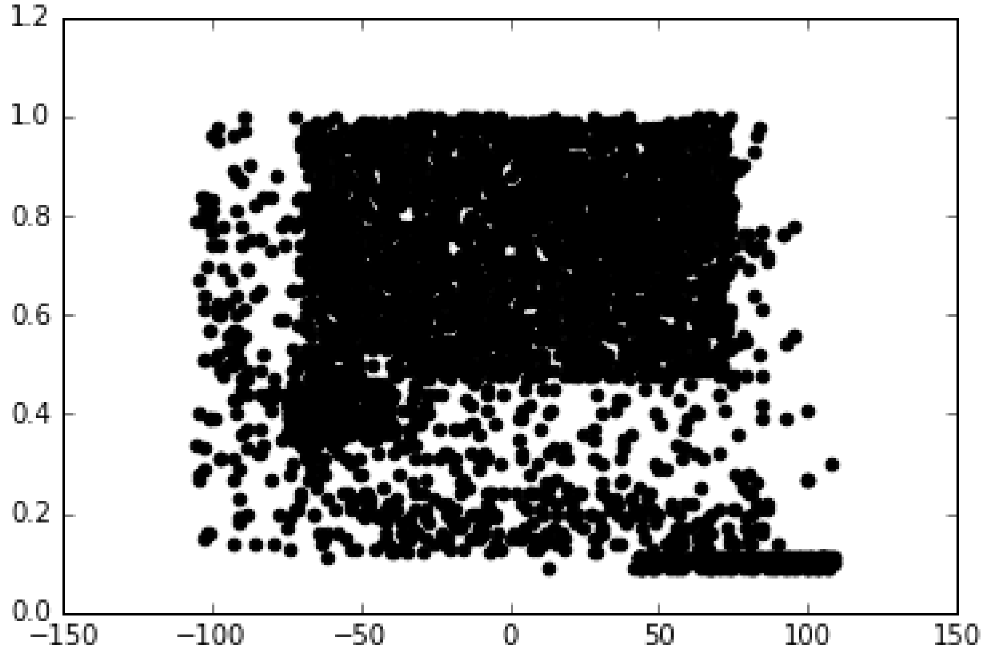
```
343                     3
2031                    4
12730                   3
7305                    2
2380                    6
8912                    3
1851                    3
9529                    2
4124                    3
11594                   7
12935                   6
14351                   4
3684                    2
14646                   3
8283                    5
12430                   6
7600                    4
10151                   2
4979                    4
7516                    3
1167                    3
11364                   3
12793                   2
12464                   5
8714                    5
6409                    5
9903                    2
4778                    3
11310                   2

[4500 rows x 4 columns]
```

```
In [30]: X_reduced = pca.transform(X_test)
         X_reduced
```

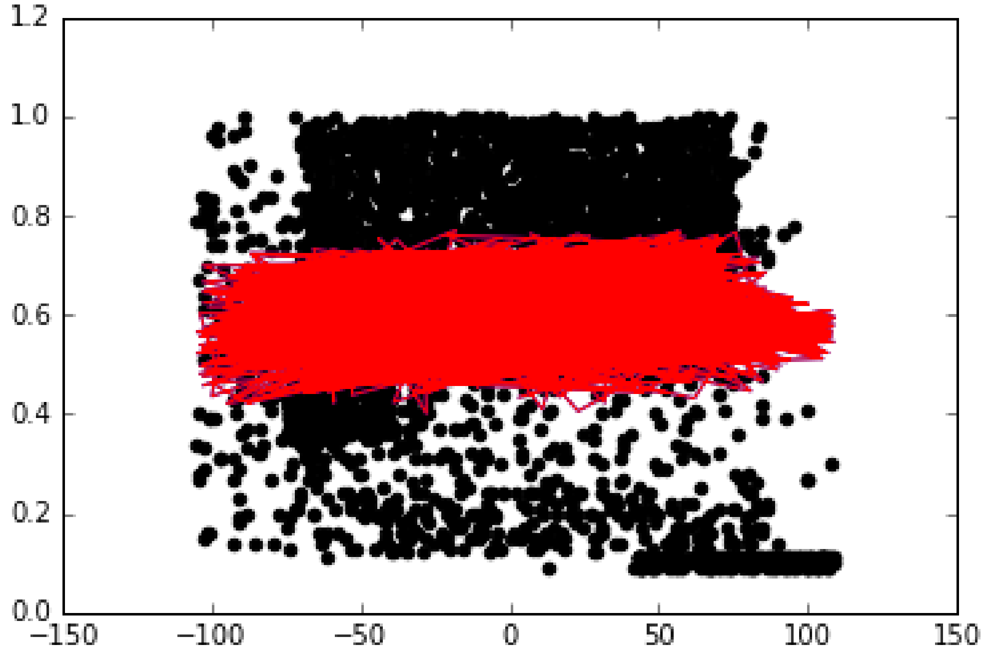
```
Out[30]: array([[ 21.96652508],
                [-64.04607826],
                [-80.03520079],
                ...,
                [ 61.94250711],
                [-59.05667198],
                [ 41.95387206]])
```

```
In [31]: plt.scatter(X_reduced, y_test,  color='black')
```

```
Out[31]: <matplotlib.collections.PathCollection at 0x7f7db1094a20>
```



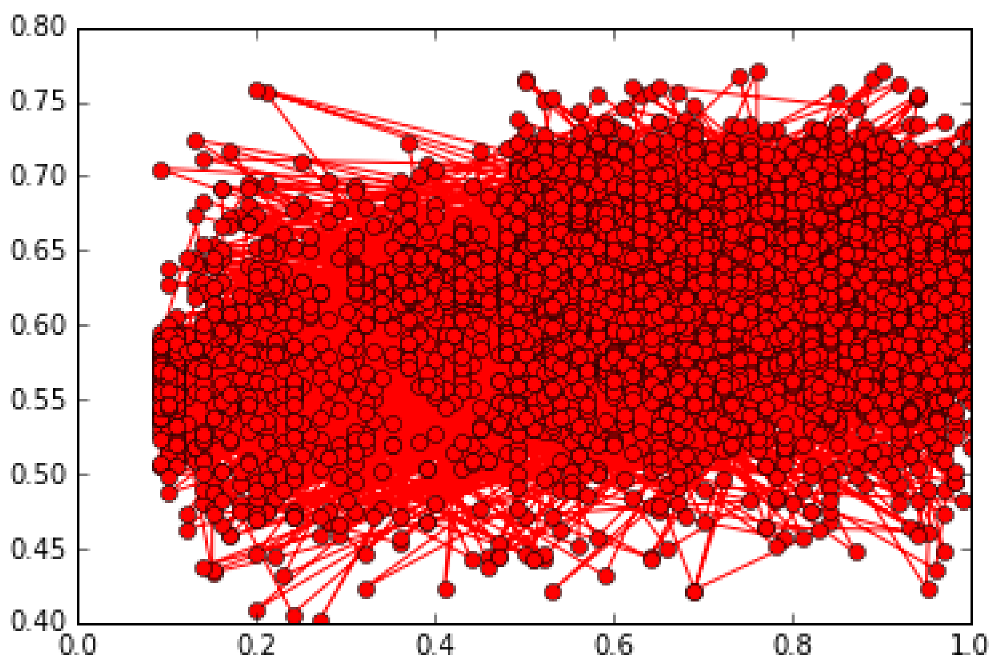
In [32]:
```
plt.scatter(X_reduced, y_test,  color='black')
plt.plot(X_reduced, test_predicted, color='blue',linewidth=1)
plt.plot(X_reduced, test_predicted, color='red',linewidth=1)

#plt.xticks(())
#plt.yticks(())

plt.show()
```



1.6 Not very insightful? Let us discuss this in class

```
In [33]: plt.plot(y_test, test_predicted, 'ro-')
```

```
Out[33]: [<matplotlib.lines.Line2D at 0x7f7db0ff12b0>]
```



In [34]:
```
np.std(np.abs(y_test-test_predicted))
```

Out[34]: 0.13005465030212102

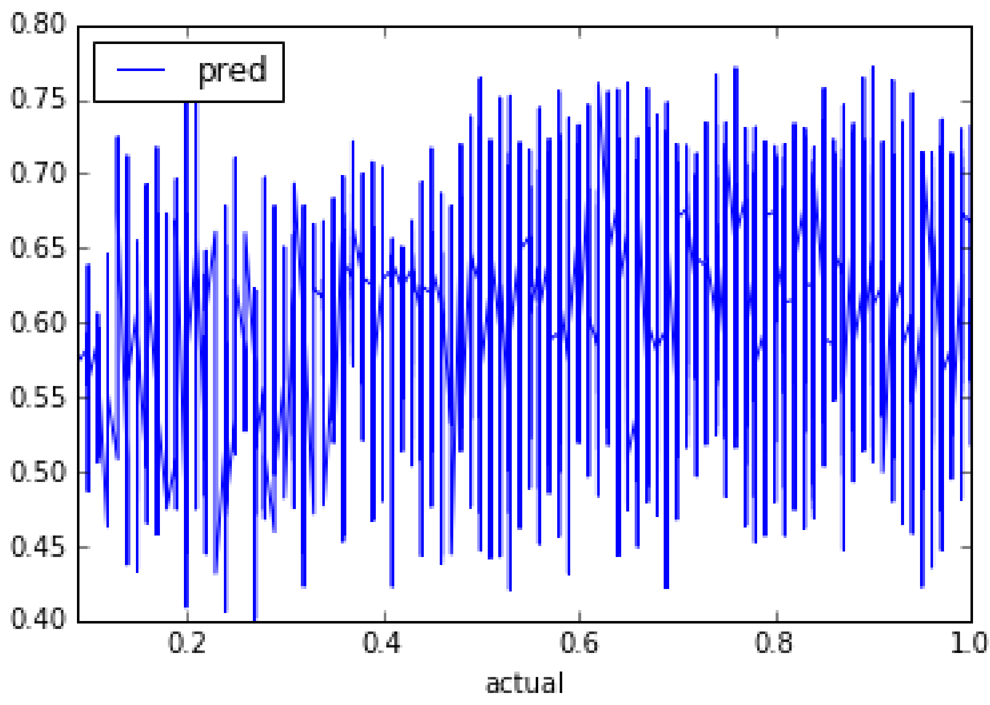
In [35]:
```
data4=pd.DataFrame({'actual':y_test,'pred':test_predicted})
data4.head()
```

Out[35]:

	actual	pred
14667	0.86	0.601516
13373	0.78	0.649397
9246	0.27	0.523364
12261	0.82	0.615714
9359	0.97	0.613623

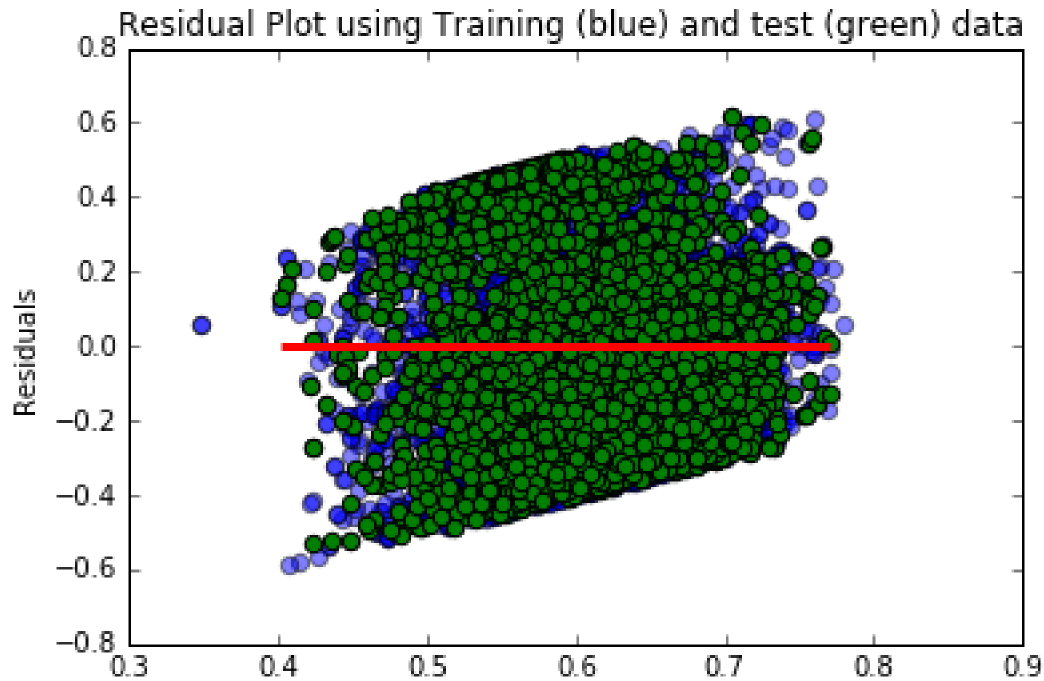
In [36]:
```
data4.sort_values('actual').plot(kind='line',x='actual',y='pred')
```

Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7db0ff1f98>



In [37]:
```
plt.scatter(reg.predict(X_train), reg.predict(X_train)-y_train,c='b',s=40,alpha=0.5)
plt.scatter(reg.predict(X_test),reg.predict(X_test)-y_test,c='g',s=40)
plt.hlines(y=0,xmin=np.min(reg.predict(X_test)),xmax=np.max(reg.predict(X_test)),color=
plt.title('Residual Plot using Training (blue) and test (green) data ')
plt.ylabel('Residuals')
```

```
Out[37]: <matplotlib.text.Text at 0x7f7db0f04438>
```




```
In [38]: data.corr()
```

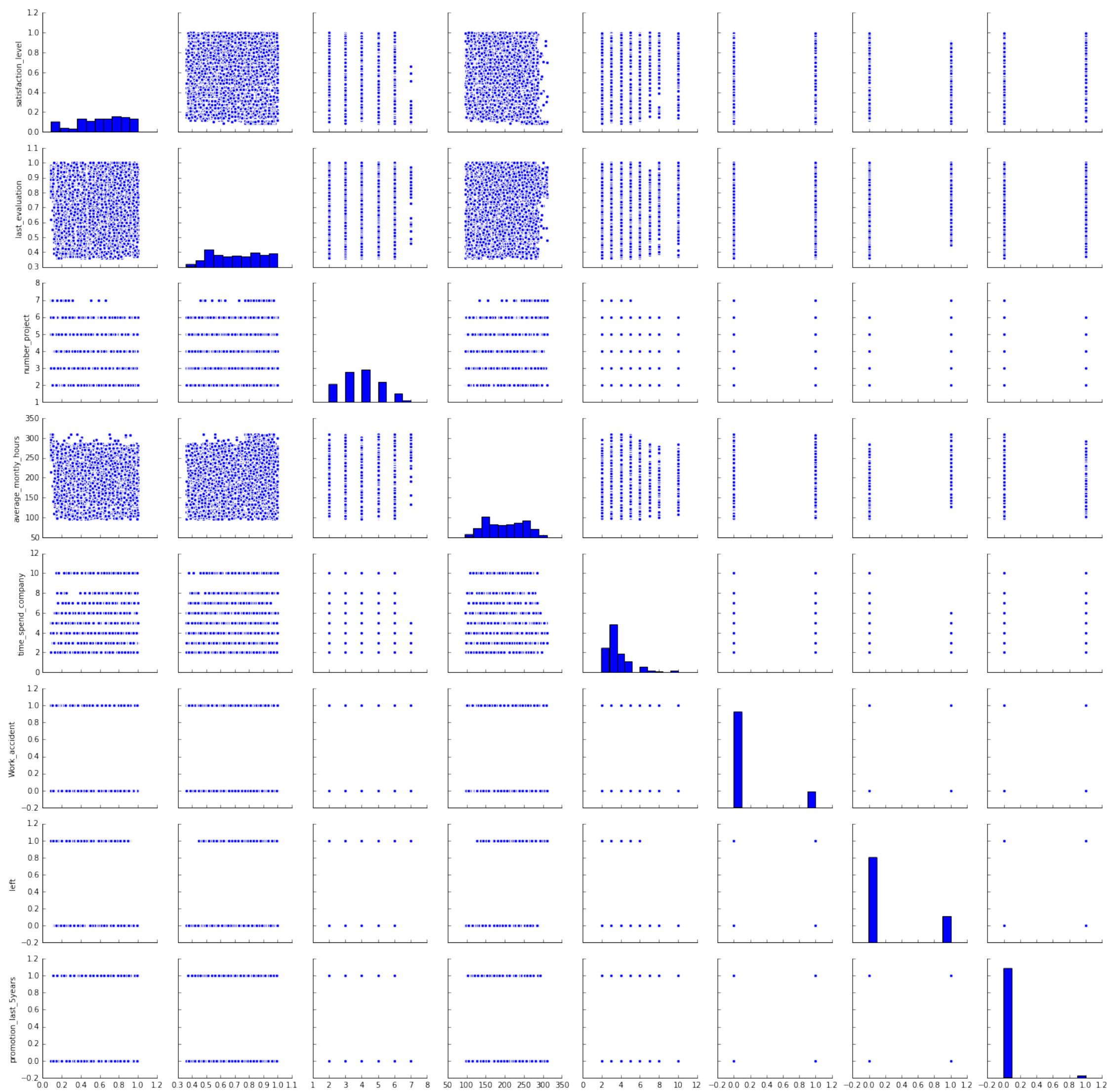
```
Out[38]:                        satisfaction_level  last_evaluation  number_project  \
         satisfaction_level              1.000000         0.105021       -0.142970
         last_evaluation                 0.105021         1.000000        0.349333
         number_project                 -0.142970         0.349333        1.000000
         average_montly_hours           -0.020048         0.339742        0.417211
         time_spend_company             -0.100866         0.131591        0.196786
         Work_accident                   0.058697        -0.007104       -0.004741
         left                           -0.388375         0.006567        0.023787
         promotion_last_5years           0.025605        -0.008684       -0.006064

                                average_montly_hours  time_spend_company  \
         satisfaction_level                -0.020048           -0.100866
         last_evaluation                    0.339742            0.131591
         number_project                     0.417211            0.196786
         average_montly_hours               1.000000            0.127755
         time_spend_company                 0.127755            1.000000
         Work_accident                     -0.010143            0.002120
         left                               0.071287            0.144822
         promotion_last_5years             -0.003544            0.067433

                                Work_accident      left  promotion_last_5years
```

satisfaction_level	0.058697	-0.388375	0.025605
last_evaluation	-0.007104	0.006567	-0.008684
number_project	-0.004741	0.023787	-0.006064
average_montly_hours	-0.010143	0.071287	-0.003544
time_spend_company	0.002120	0.144822	0.067433
Work_accident	1.000000	-0.154622	0.039245
left	-0.154622	1.000000	-0.061788
promotion_last_5years	0.039245	-0.061788	1.000000

In [39]:
```
import seaborn as sns
```

In [40]:
```
sns.pairplot(data)
```

Out[40]: <seaborn.axisgrid.PairGrid at 0x7f7db0ef4128>



In [42]:
```
rng = np.random.RandomState(1)
x = 10 * rng.rand(50)
y = 2 * x - 5 + rng.randn(50)
plt.scatter(x, y);
```