Prediction

Linear regression with one attribute

In [1]:

```
# Load packages
import pandas as pd
from sklearn.linear_model import LinearRegression
% matplotlib inline
```

In [2]:

```
# Read data
df = pd.read_csv('rent.csv')
```

In [3]:

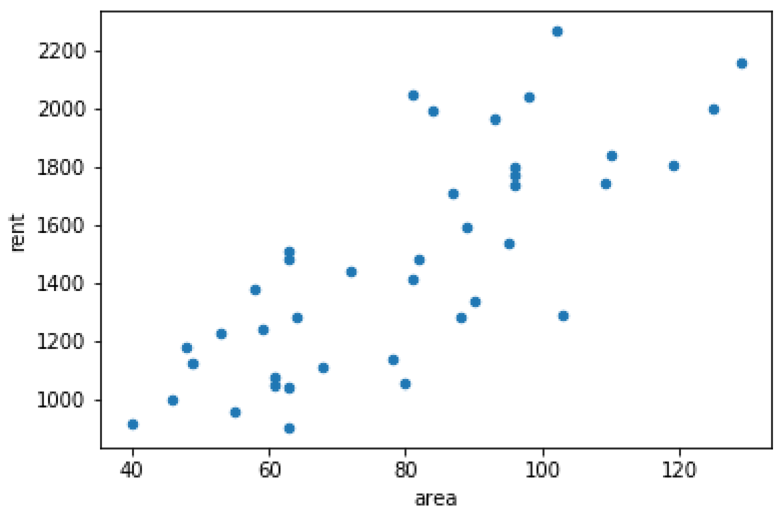
```
# Inspect data set
df.head()
```

Out[3]:

	area	rent
0	58	1380
1	72	1440
2	55	960
3	129	2160
4	78	1134

In [4]:

```
# Plot data
df.plot.scatter(x='area', y='rent');
```



In [5]:

```
# Prepare data
X = df[['area']]
y = df['rent']
```

In [6]:

```
print(X.shape)
print(y.shape)
```

```
(40, 1)
(40,)
```

In [7]:

```
# Fit model to data
model = LinearRegression()
model.fit(X.values, y.values)
```

Out[7]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [8]:

```
# Display intercept
model.intercept_
```

Out[8]:

```
419.14207846231034
```

In [9]:

```
# Display beta
model.coef_
```

Out[9]:

```
array([13.17119702])
```

In [10]:

```
# Make predictions
predictions = model.predict(X)
predictions[:5]
```

Out[10]:

```
array([1183.07150541, 1367.46826364, 1143.55791437, 2118.22649358,
       1446.49544574])
```

In [11]:

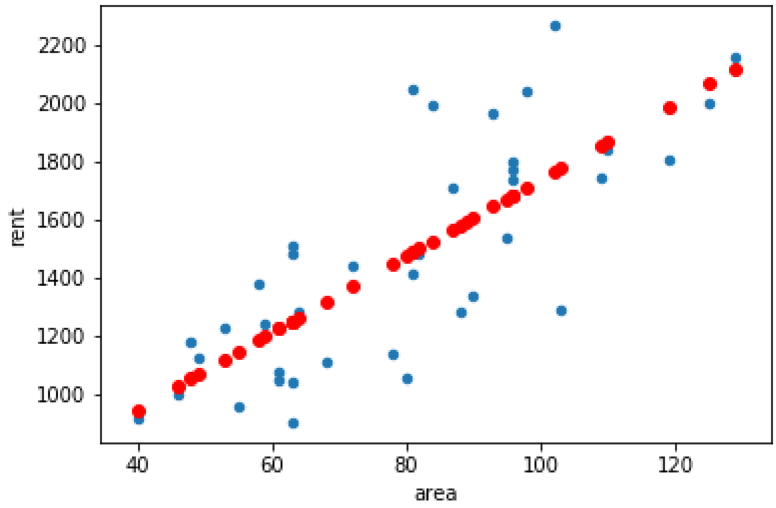
```
# Display R^2
model.score(X, y)
```

Out[11]:

```
0.600629022747489
```

In [12]:

```
# Plot regression line
ax = df.plot.scatter(x='area', y='rent')
ax.scatter(df.area, predictions, color='red');
```



Linear regression with multiple attributes

In [13]:

```
# Load another data set
df = pd.read_csv('rent_extended.csv')
df.head()
```

Out[13]:

	area	rent	neighborhood	age
0	58	1380	wabern	20
1	72	1440	laenggasse	43
2	55	960	ostring	42
3	129	2160	ostring	1
4	78	1134	ostring	47

In [14]:

```
# Prepare data
X = pd.get_dummies(df, columns=['neighborhood'], drop_first=True).drop('rent', axis=1)
y = df['rent']
```

In [15]:

```
X.head()
```

Out[15]:

	area	age	neighborhood_ostring	neighborhood_wabern
0	58	20	0	1
1	72	43	0	0
2	55	42	1	0
3	129	1	1	0
4	78	47	1	0

In [16]:

```
# Fit model to data
model = LinearRegression()
model.fit(X.values, y.values)
```

Out[16]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [17]:

```
# Display intercept
model.intercept_
```

Out[17]:

```
525.7064975636313
```

In [18]:

```
# Display betas
pd.Series(model.coef_, index=X.columns)
```

Out[18]:

```
area                      13.567127
age                       -0.542951
neighborhood_ostring    -150.189523
neighborhood_wabern     -189.035836
dtype: float64
```

In [19]:

```
# Make predictions
predictions = model.predict(X)
predictions[:5]
```

Out[19]:

```
array([1112.70503118, 1479.19278082, 1098.90504372, 2125.13345303,
       1408.2342187 ])
```

Polynomial regression

In [20]:

```
from sklearn.preprocessing import PolynomialFeatures
import numpy as np
```

In [21]:

```
# Prepare data
X = df[['area']].copy()
X['area^2'] = df.area**2
X['area^3'] = df.area**3
y = df['rent']
```

In [22]:

```
X.head()
```

Out[22]:

	area	area^2	area^3
0	58	3364	195112
1	72	5184	373248
2	55	3025	166375
3	129	16641	2146689
4	78	6084	474552

In [23]:

```
# Prepare data: alternative
poly = PolynomialFeatures(3, include_bias=False)
values = poly.fit_transform(df[['area']])
names = poly.get_feature_names(['area'])
X = pd.DataFrame(values, columns=names)
y = df['rent']
```

In [24]:

```
X.head()
```

Out[24]:

	area	area^2	area^3
0	58.0	3364.0	195112.0
1	72.0	5184.0	373248.0
2	55.0	3025.0	166375.0
3	129.0	16641.0	2146689.0
4	78.0	6084.0	474552.0

In [25]:

```
# Fit model to data
model = LinearRegression()
model.fit(X.values, y.values)
```

Out[25]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [26]:

```
# Display betas
pd.Series(model.coef_, index=X.columns)
```

Out[26]:

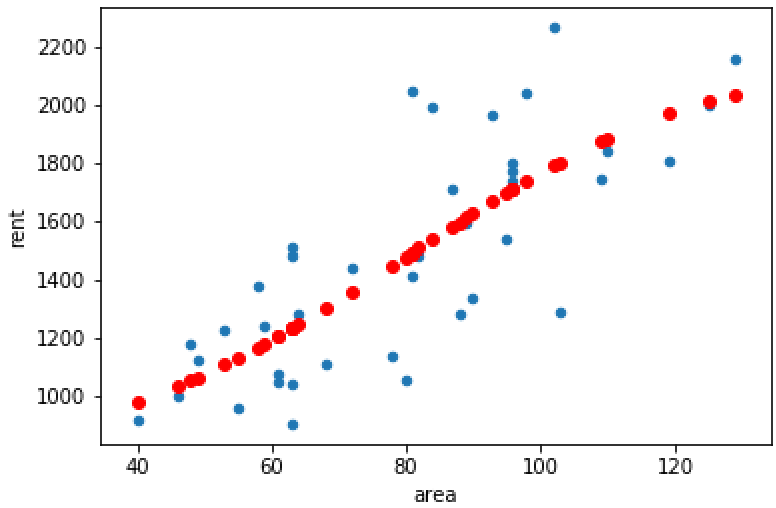
```
area      -14.494117
area^2      0.365917
area^3     -0.001519
dtype: float64
```

In [27]:

```
# Make predictions
predictions = model.predict(X)
```

In [28]:

```
# Plot predictions
ax = df.plot.scatter(x='area', y='rent')
ax.scatter(df.area, predictions, color='red');
```



Inference statistics for linear regression

In [29]:

```
# Optional: get OLS regression results
from statsmodels.api import OLS
from patsy import dmatrices
```

In [30]:

```
# Define multivariate regression model
y, X = dmatrices('rent ~ area + neighborhood + age', df, return_type='dataframe')
```

In [31]:

```
# Define polynomial regression model
y, X = dmatrices('rent ~ area + I(area**2) + I(area**3)', df, return_type='dataframe')
```

In [32]:

```
OLS(y, X).fit().summary()
```

Out[32]:

OLS Regression Results

Dep. Variable:	rent	**R-squared:**	0.605
Model:	OLS	**Adj. R-squared:**	0.572
Method:	Least Squares	**F-statistic:**	18.35
Date:	Fri, 12 Oct 2018	**Prob (F-statistic):**	2.16e-07
Time:	09:51:26	**Log-Likelihood:**	-275.68
No. Observations:	40	**AIC:**	559.4
Df Residuals:	36	**BIC:**	566.1
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1071.8307	1613.838	0.664	0.511	-2201.184	4344.845
area	-14.4941	62.012	-0.234	0.817	-140.260	111.272
I(area ** 2)	0.3659	0.757	0.484	0.632	-1.168	1.900
I(area ** 3)	-0.0015	0.003	-0.515	0.609	-0.007	0.004

Omnibus:	0.469	**Durbin-Watson:**	2.370
Prob(Omnibus):	0.791	**Jarque-Bera (JB):**	0.353
Skew:	0.220	**Prob(JB):**	0.838
Kurtosis:	2.864	**Cond. No.**	3.29e+07

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.29e+07. This might indicate that there are
strong multicollinearity or other numerical problems.

In [ ]:

```

```