ml-assignment-4016

April 12, 2023

1 Simple Linear Regression

[134]:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

[135]:

```
df = pd.read_csv('/content/placement.csv')
df.head()
```

[135]:

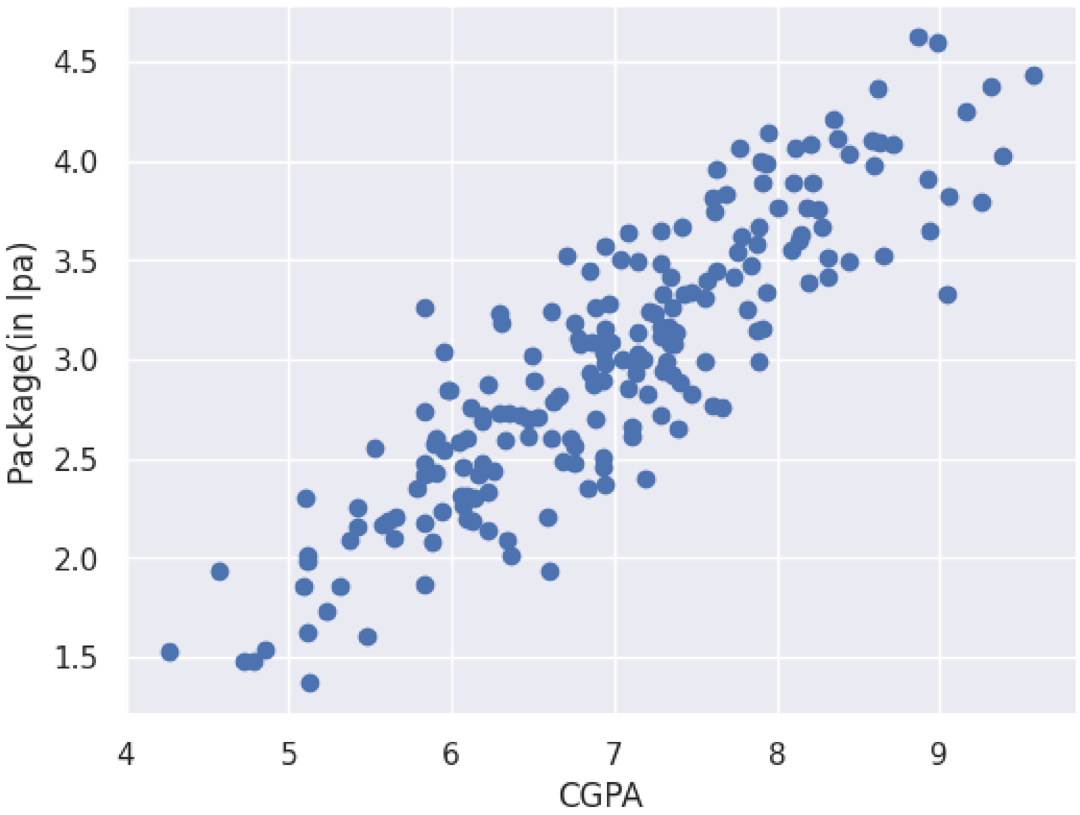
```
   cgpa  package
0  6.89     3.26
1  5.12     1.98
2  7.82     3.25
3  7.42     3.67
4  6.94     3.57
```

[136]:

```
plt.scatter(df['cgpa'],df['package'])
plt.xlabel('CGPA')
plt.ylabel('Package(in lpa)')
```

[136]:

```
Text(0, 0.5, 'Package(in lpa)')
```



[137]:
```
df.describe()
```

[137]:
```
            cgpa     package
count  200.000000  200.000000
mean     6.990500    2.996050
std      1.069409    0.691644
min      4.260000    1.370000
25%      6.190000    2.487500
50%      6.965000    2.995000
75%      7.737500    3.492500
max      9.580000    4.620000
```

[138]:
```
X = df.iloc[:,0:1]
y = df.iloc[:,-1]
```

[139]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

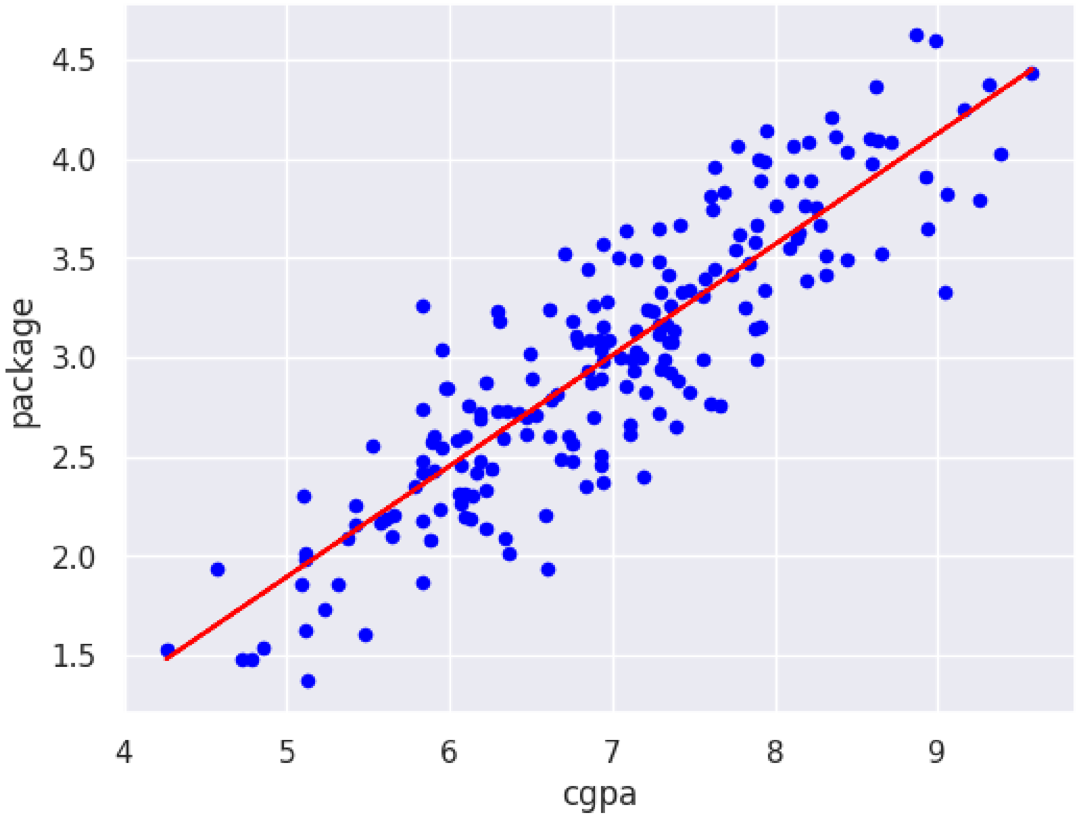
[140]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
```

[140]:
```
LinearRegression()
```

[141]:
```
model.coef_, model.intercept_
```

[141]:
```
(array([0.55795197]), -0.8961119222429144)
```

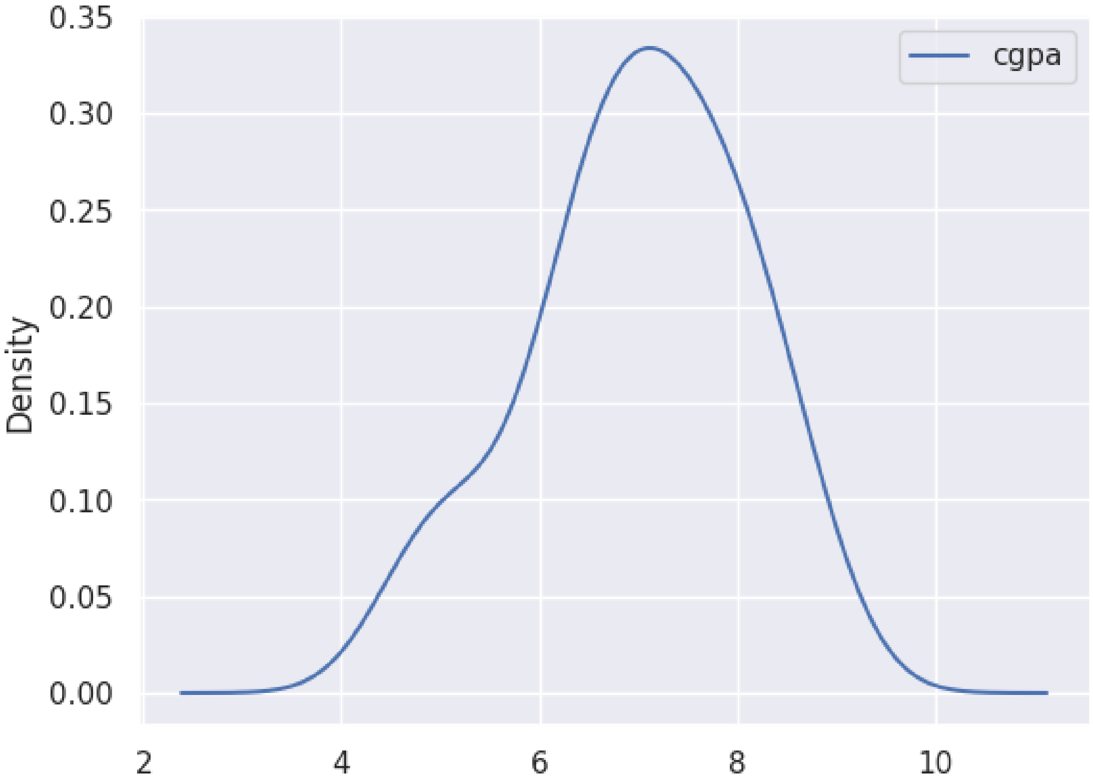
[142]:
```
df.plot.scatter('cgpa', 'package', color='blue')
plt.plot(
    np.array(df['cgpa']),
    model.coef_[0] * np.array(df['cgpa']) + model.intercept_,
    color='red'
)
plt.show()
```



[143]:
```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.
 ↪2,random_state=2)
```

[144]:
```
X_test.plot.density()
plt.show()
```



[145]:
```
model.predict(X_test.iloc[0].values.reshape(1,1))
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:439: UserWarning:

X does not have valid feature names, but LinearRegression was fitted with
feature names
```

[145]:
```
array([3.89111601])
```

[147]:
```
m = model.coef_
b = model.intercept_
print(m,b)
```

```
[0.55795197] -0.8961119222429144
```

2 Linear Regression with Multiple Variable

[105]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
sns.set_theme(style='darkgrid')
```

[92]:

```
df = pd.read_csv('/content/houseprices2.csv')
df.head()
```

[92]:

```
   area  bedrooms  age   price
0  2600       3.0   20  550000
1  3000       4.0   15  565000
2  3200       NaN   18  610000
3  3600       3.0   30  595000
4  4000       5.0    8  760000
```

[93]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   area      6 non-null      int64
 1   bedrooms  5 non-null      float64
 2   age       6 non-null      int64
 3   price     6 non-null      int64
dtypes: float64(1), int64(3)
memory usage: 320.0 bytes
```

[94]:

```
df.describe()
```

[94]:

```
              area  bedrooms        age          price
count     6.000000   5.00000   6.000000       6.000000
mean   3416.666667   4.20000  16.500000  648333.333333
std     587.934237   1.30384   8.288546  109117.673484
min    2600.000000   3.00000   8.000000  550000.000000
25%    3050.000000   3.00000   9.750000  572500.000000
50%    3400.000000   4.00000  16.500000  602500.000000
75%    3900.000000   5.00000  19.500000  722500.000000
max    4100.000000   6.00000  30.000000  810000.000000
```

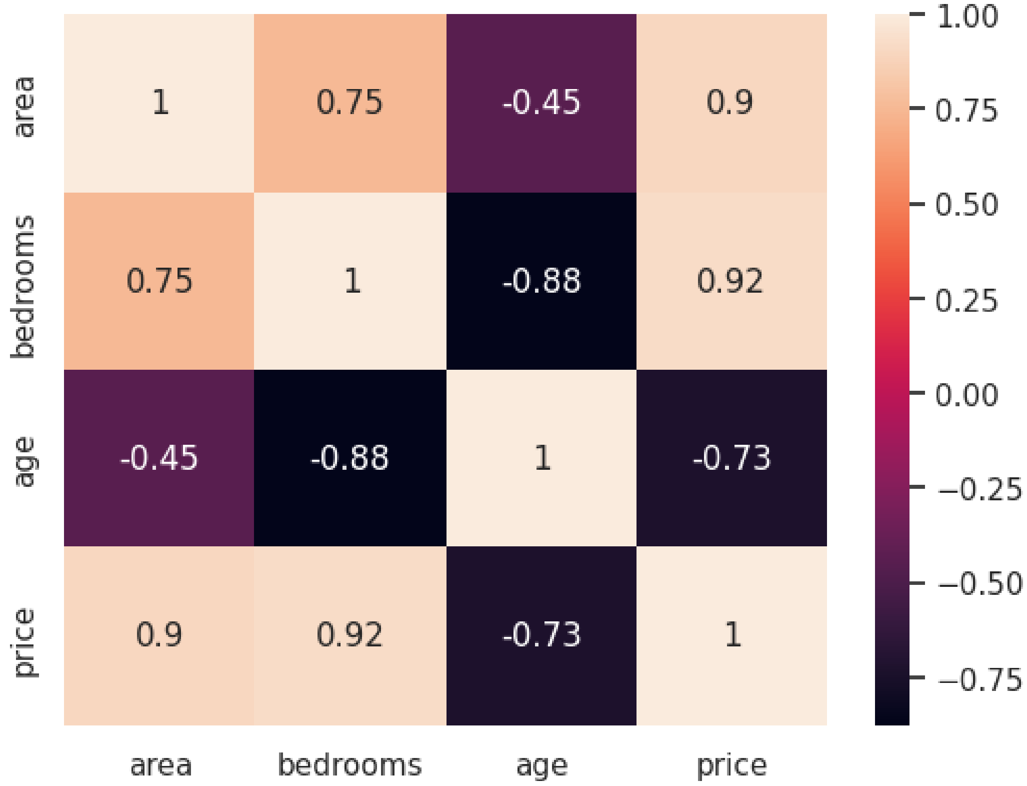
[106]:

```
fig = px.scatter_3d(df, x='area', y='bedrooms', z='age')
```

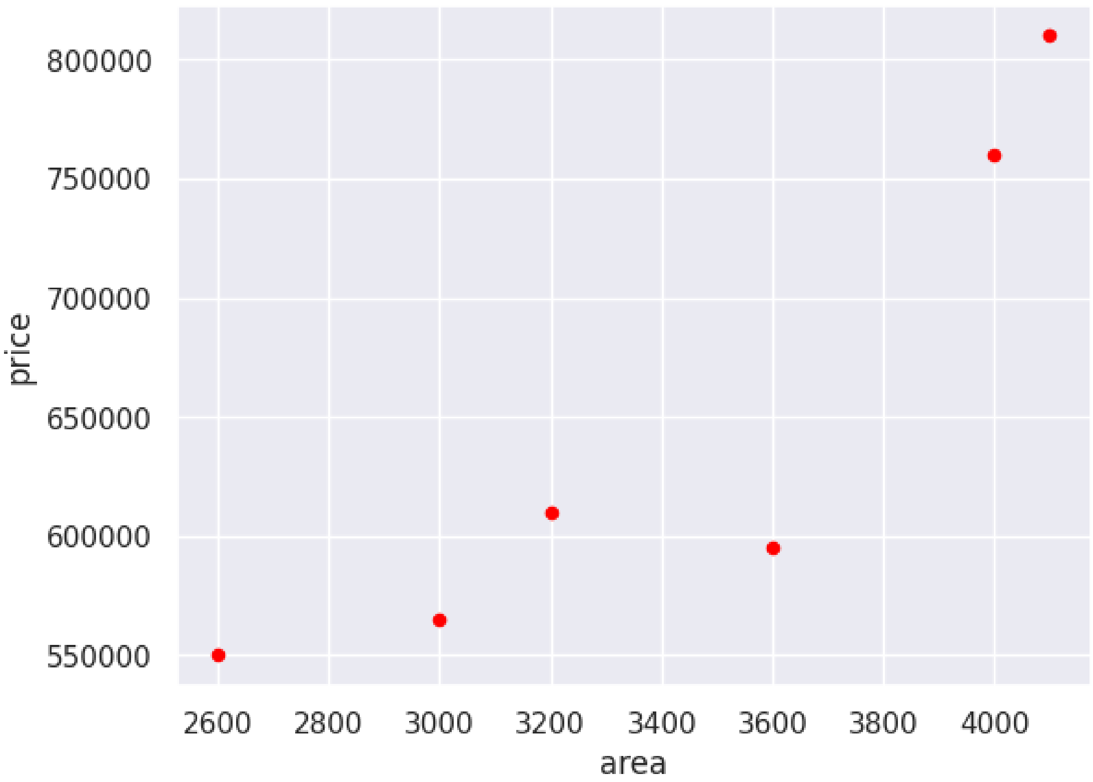
```
fig.show()
```

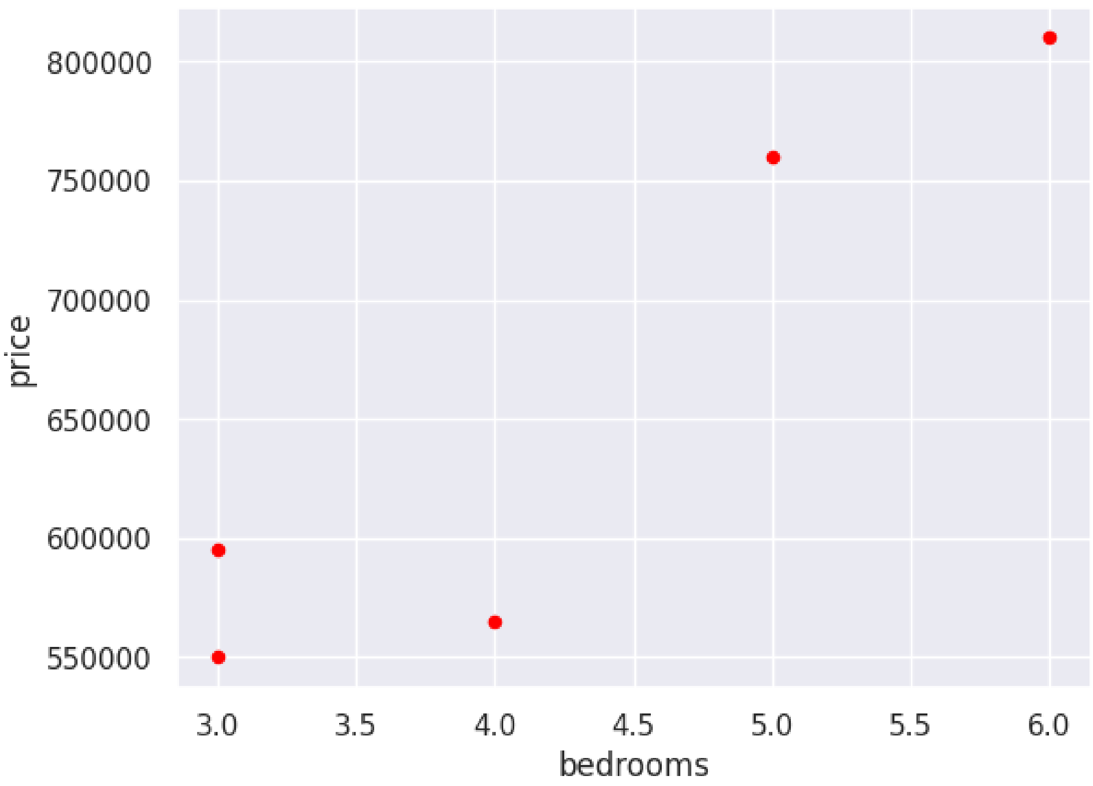
[95]:
```
sns.heatmap(df.corr(), annot=True)
```

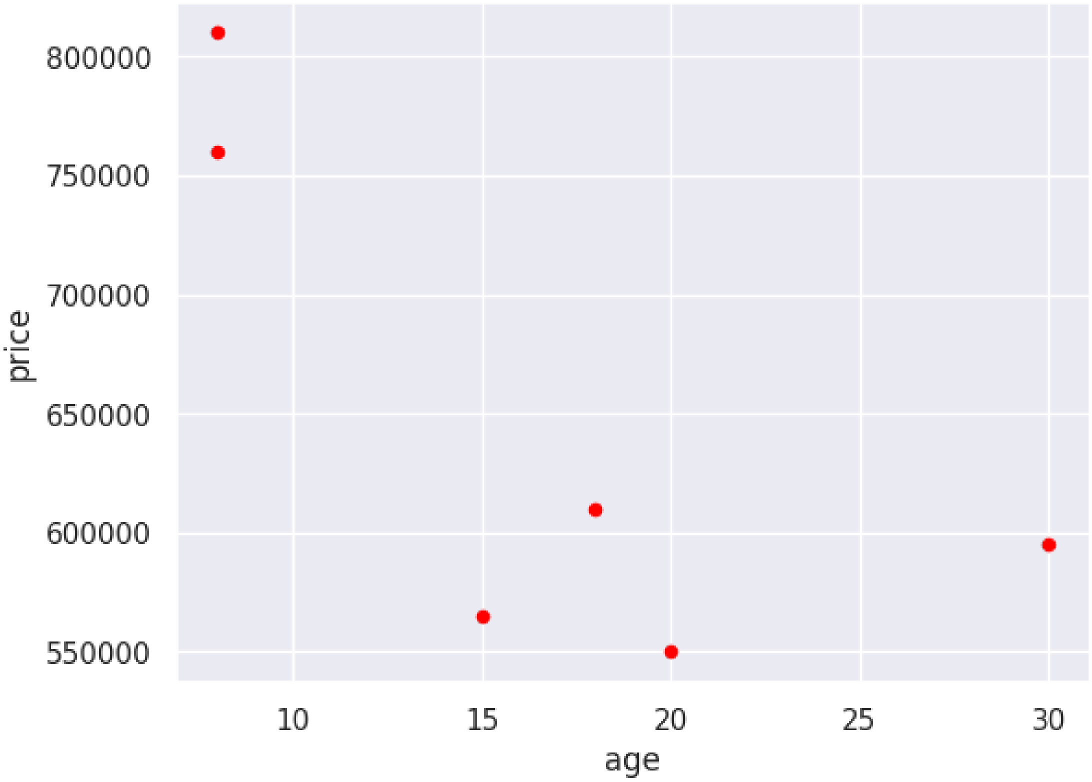
[95]:
```
<Axes: >
```



[96]:
```
for x in df.columns[:-1]:
    df.plot.scatter(x, 'price', color='red')
    plt.show()
```


800000
750000
700000
650000
600000
550000
price
2600
2800
3000
3200
3400
3600
3800
4000
area

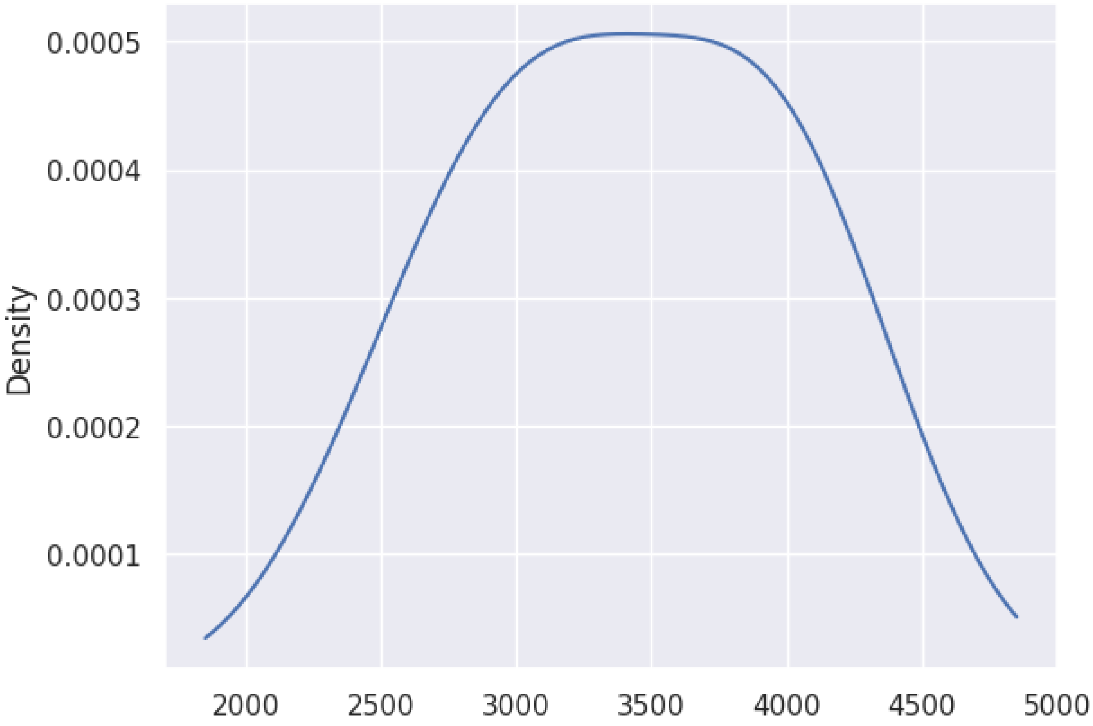

800000
750000
700000
650000
600000
550000
price
3.0
3.5
4.0
4.5
5.0
5.5
6.0
bedrooms

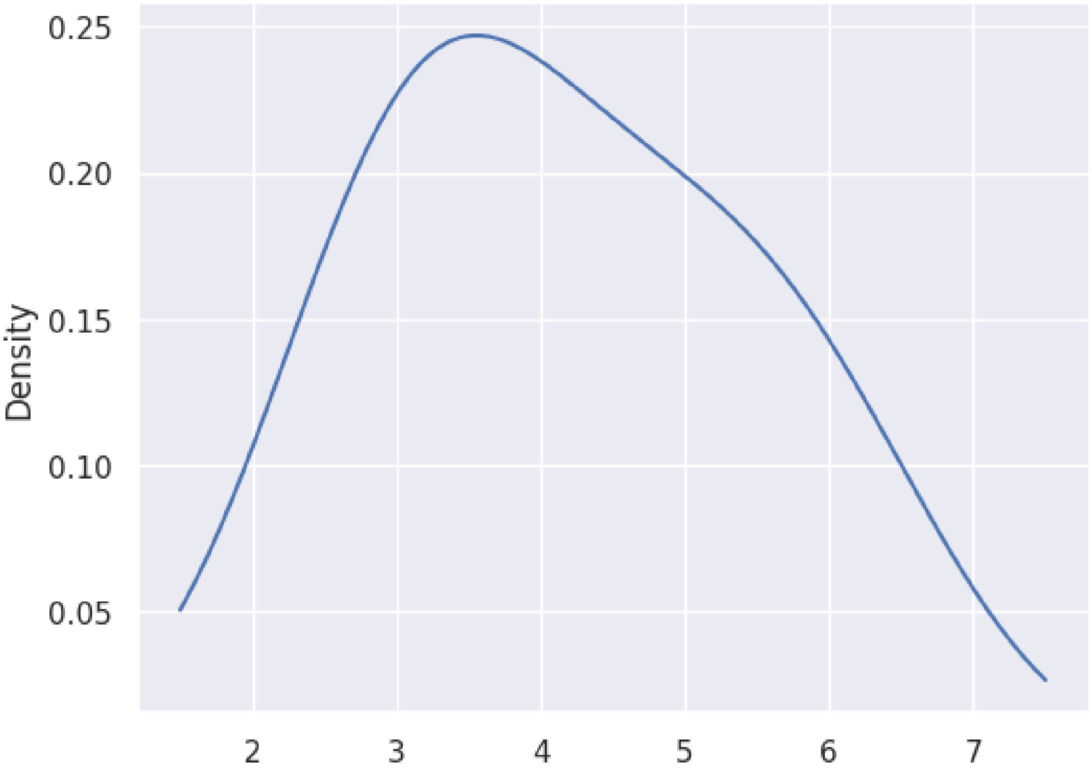


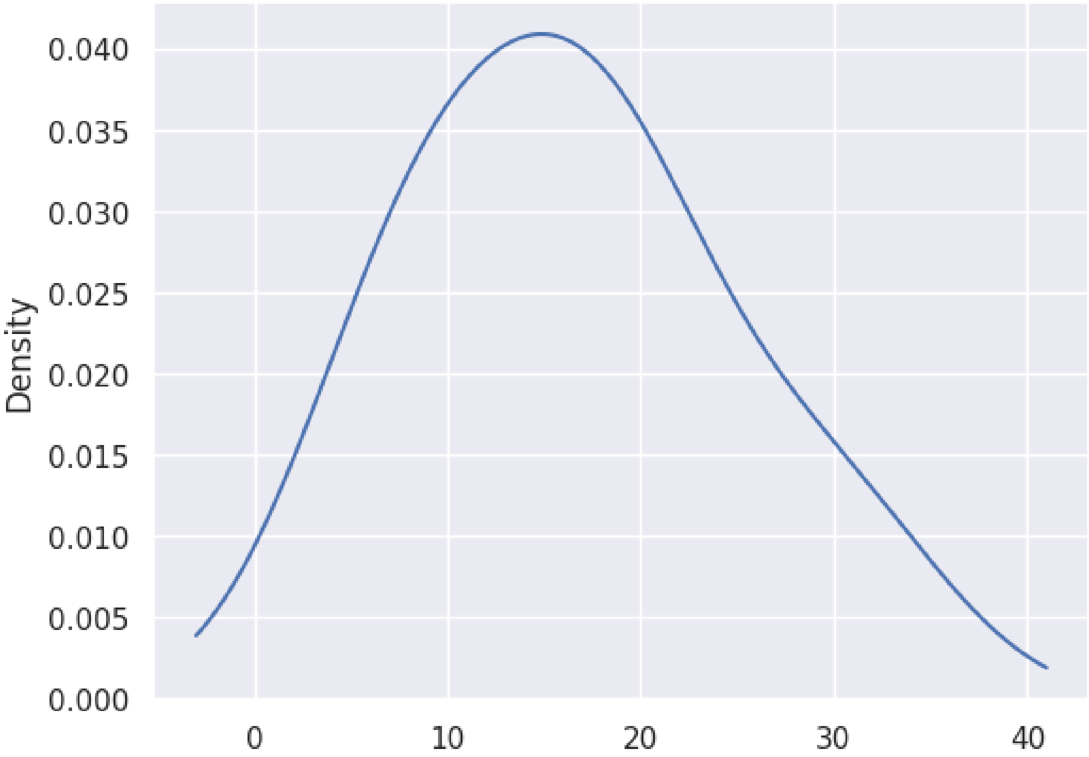
[97]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()

lr.fit(X_train,y_train)
```

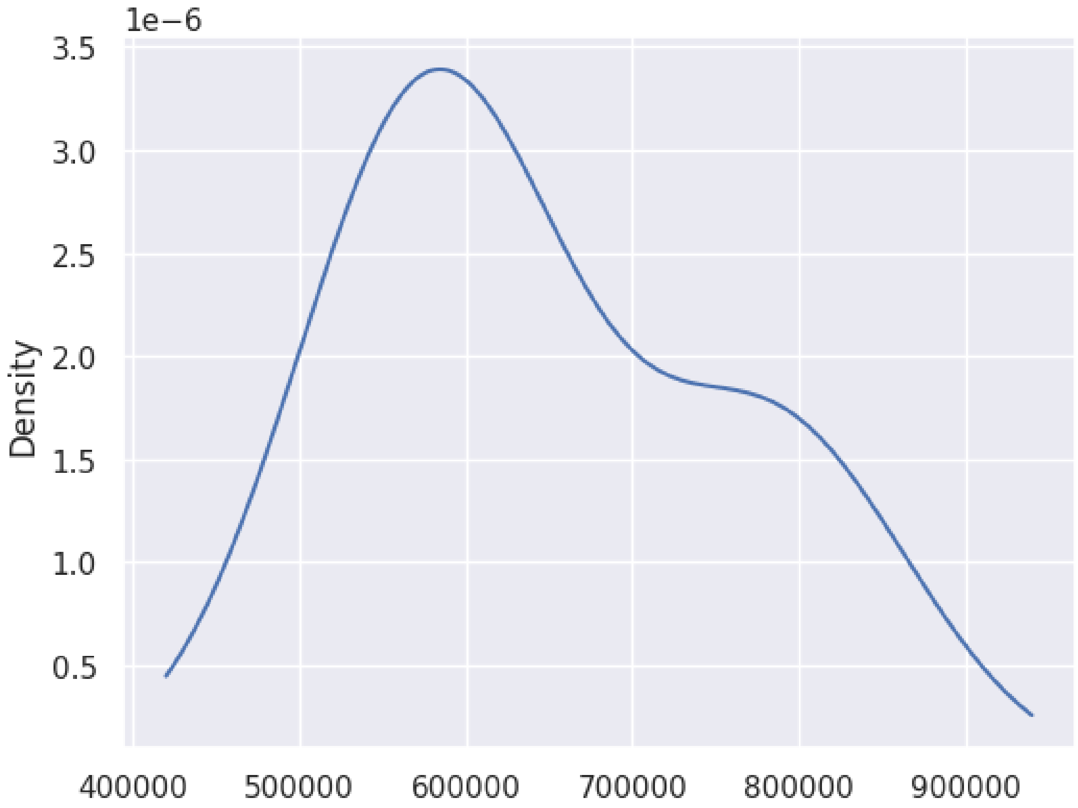
[97]:
```
LinearRegression()
```

[98]:
```
for x in df.columns:
    df[x].plot.density()
    plt.show()
```


0.0005
0.0004
0.0003
0.0002
0.0001
Density
2000
2500
3000
3500
4000
4500
5000


0.25
0.20
0.15
0.10
0.05
Density
2
3
4
5
6
7


Density
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
10
20
30
40



[101]:
```
df['bedrooms'] = df['bedrooms'].fillna(
  df['bedrooms'].median()
)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   area      6 non-null      int64
 1   bedrooms  6 non-null      float64
 2   age       6 non-null      int64
 3   price     6 non-null      int64
dtypes: float64(1), int64(3)
memory usage: 320.0 bytes
```

[102]:
```
X_train = df.drop('price', axis=1)
y_train = df['price']
```

[103]:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
```

[103]:

```
LinearRegression()
```

[104]:

```
model.coef_, model.intercept_
```

[104]:

```
(array([  112.06244194, 23388.88007794, -3231.71790863]), 221323.00186540396)
```

3 Polynomial Regression

[132]:

```
import math
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(style='darkgrid')
```

[148]:

```
df = pd.read_csv('/content/Position_Salaries.csv')
df
```

[148]:

```
            Position  Level   Salary
0   Business Analyst      1    45000
1  Junior Consultant      2    50000
2  Senior Consultant      3    60000
3            Manager      4    80000
4    Country Manager      5   110000
5     Region Manager      6   150000
6            Partner      7   200000
7     Senior Partner      8   300000
8            C-level      9   500000
9                CEO     10  1000000
```

[149]:

```
df.describe()
```

[149]:

```
          Level          Salary
count  10.00000       10.000000
mean    5.50000   249500.000000
std     3.02765   299373.883668
min     1.00000    45000.000000
25%     3.25000    65000.000000
50%     5.50000   130000.000000
75%     7.75000   275000.000000
max    10.00000  1000000.000000
```

[150]:

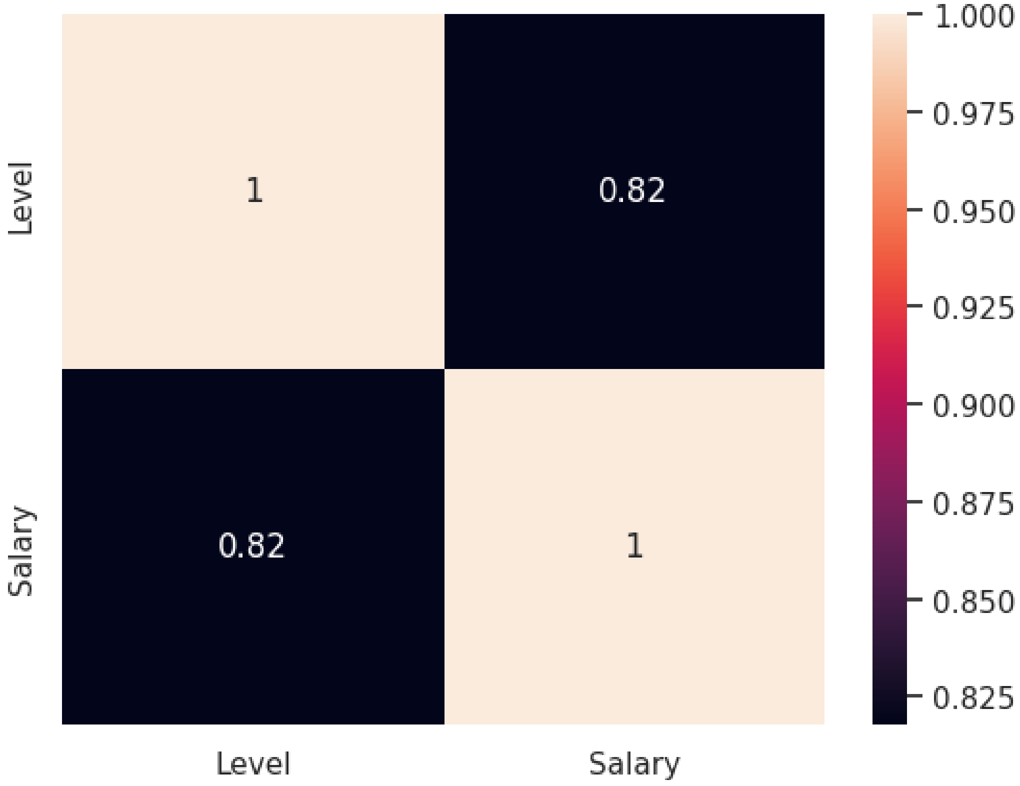
```
df = df.drop('Position', axis=1)
df.head()
```

[150]:

```
   Level  Salary
0      1   45000
1      2   50000
2      3   60000
3      4   80000
4      5  110000
```

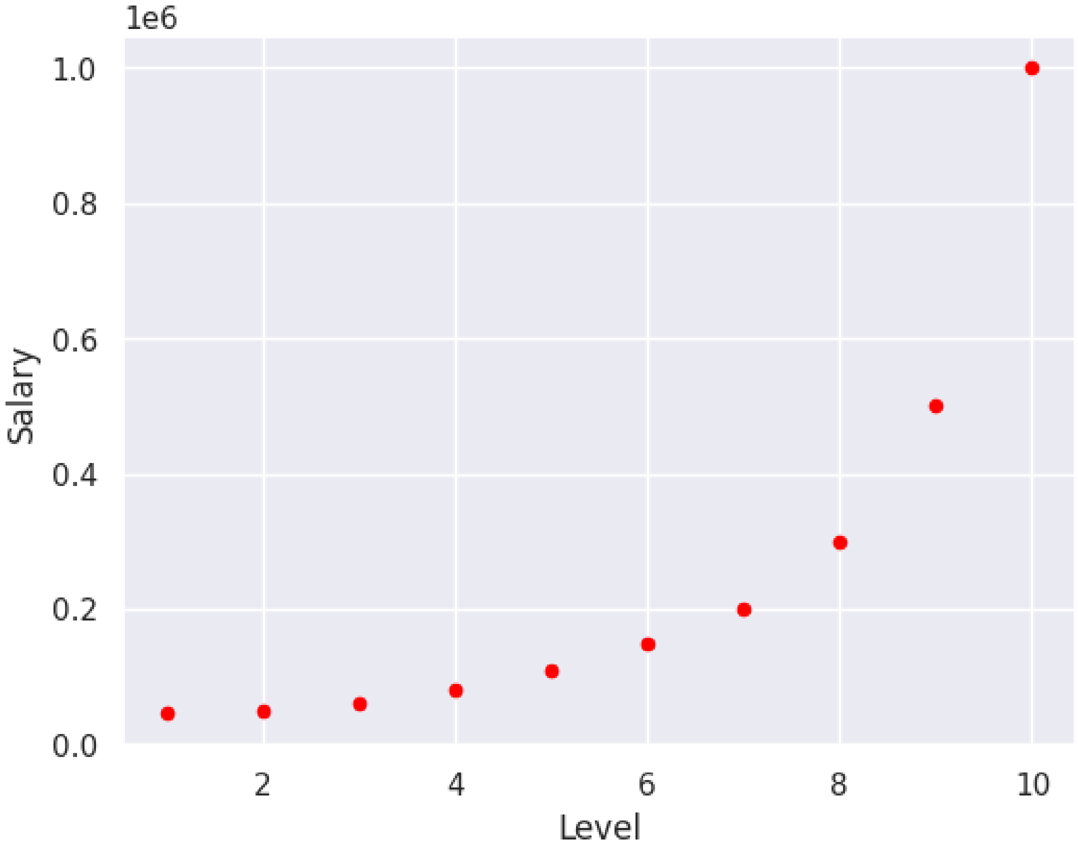
[151]:

```
sns.heatmap(df.corr(), annot=True)
plt.show()
```



[152]:

```
df.plot.scatter('Level', 'Salary', color='red')
plt.show()
```



[153]:
```
X_train = df.drop('Salary', axis=1)
y_train = df['Salary']
```

- 1 Degree

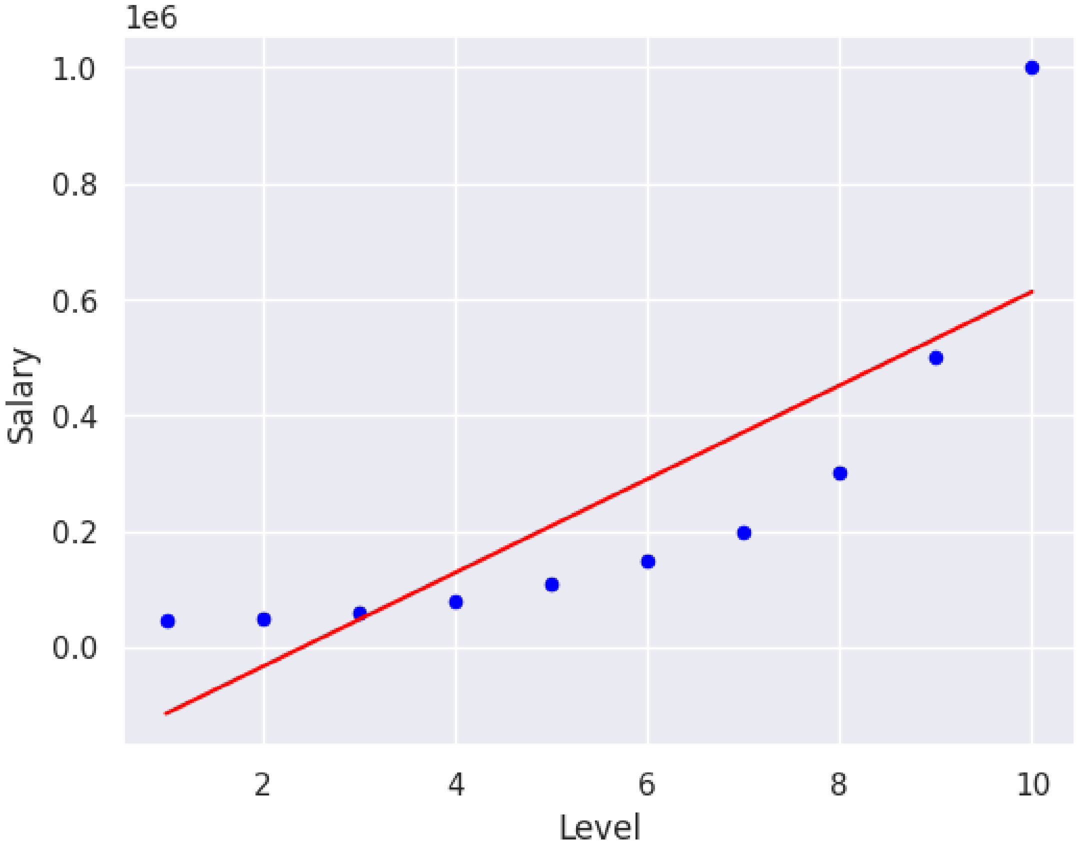
[154]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
```

[154]:
```
LinearRegression()
```

[155]:
```
model.coef_, model.intercept_
```

[155]:
```
(array([80878.78787879]), -195333.33333333337)
```

[156]:
```
df.plot.scatter('Level', 'Salary', color='blue')
plt.plot(
    np.array(df['Level']),
    model.coef_[0] * np.array(df['Level']) + model.intercept_,
    color='red'
)
plt.show()
```



- 2 Degree

[157]:
```
from sklearn.preprocessing import PolynomialFeatures
quadratic = PolynomialFeatures(degree=2)
X_deg2 = quadratic.fit_transform(X_train)
X_deg2
```

[157]:
```
array([[  1.,   1.,   1.],
       [  1.,   2.,   4.],
       [  1.,   3.,   9.],
       [  1.,   4.,  16.],
       [  1.,   5.,  25.],
       [  1.,   6.,  36.],
       [  1.,   7.,  49.],
       [  1.,   8.,  64.],
       [  1.,   9.,  81.],
       [  1.,  10., 100.]])
```

[158]:

```
model_deg2 = LinearRegression()
model_deg2.fit(X_deg2, y_train)
```

[158]:

```
LinearRegression()
```

[159]:

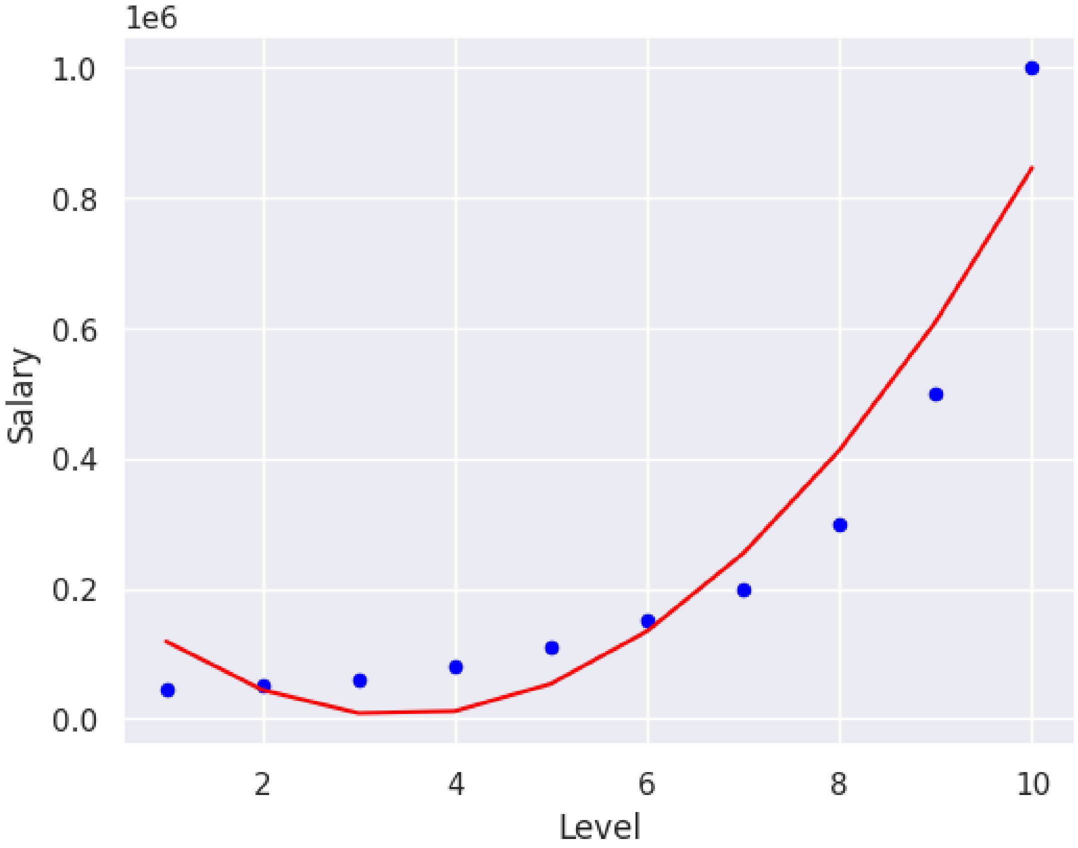
```
model_deg2.coef_, model_deg2.intercept_
```

[159]:

```
(array([      0.        , -132871.21212121,   19431.81818182]),
 232166.6666666664)
```

[160]:

```
df.plot.scatter('Level', 'Salary', color='blue')
plt.plot(
    np.array(df['Level']),
    model_deg2.coef_[1] * np.array(df['Level'])
        + model_deg2.coef_[2] * np.array(df['Level']) ** 2
        + model_deg2.intercept_,
    color='red'
)
plt.show()
```



- 3 Degree

[161]:
```
cubic = PolynomialFeatures(degree=3)
X_deg3 = cubic.fit_transform(X_train)
X_deg3
```

[161]:
```
array([[   1.,    1.,    1.,    1.],
       [   1.,    2.,    4.,    8.],
       [   1.,    3.,    9.,   27.],
       [   1.,    4.,   16.,   64.],
       [   1.,    5.,   25.,  125.],
       [   1.,    6.,   36.,  216.],
       [   1.,    7.,   49.,  343.],
       [   1.,    8.,   64.,  512.],
       [   1.,    9.,   81.,  729.],
       [   1.,   10.,  100., 1000.]])
```

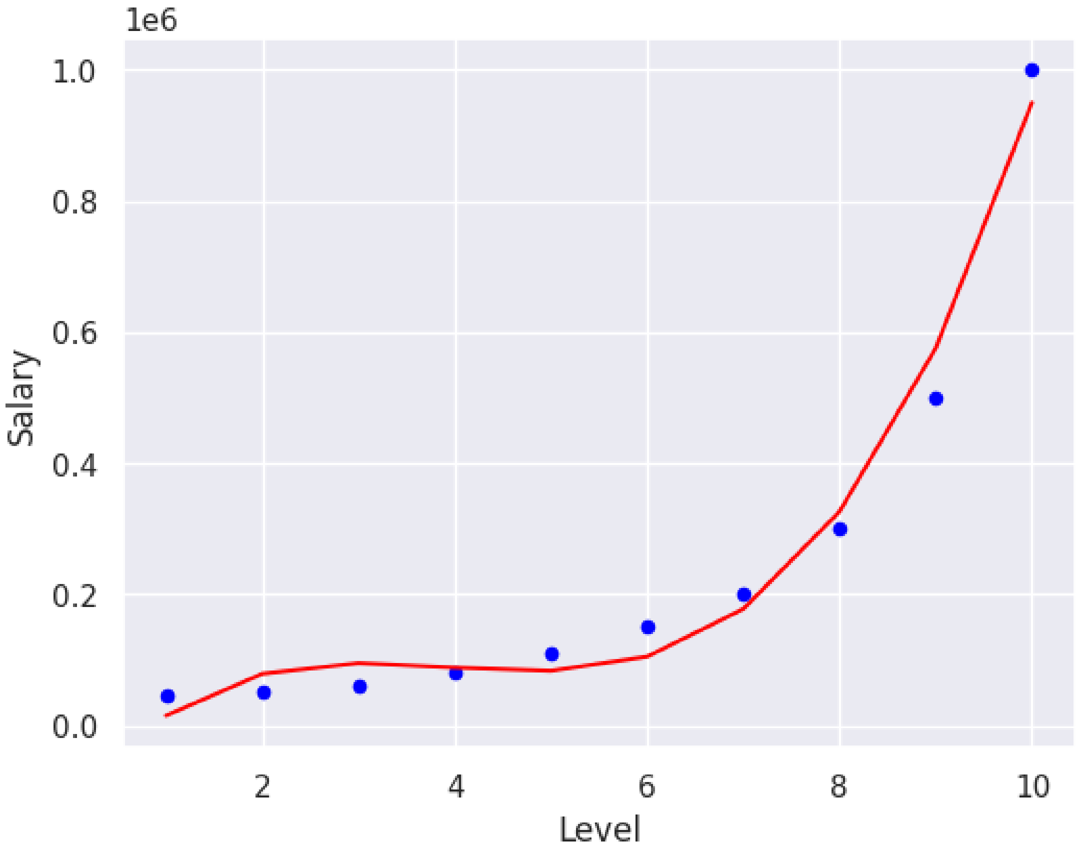
[162]:
```
model_deg3 = LinearRegression()
model_deg3.fit(X_deg3, y_train)
```

[162]:
```
LinearRegression()
```

[163]:
```
model_deg3.coef_, model_deg3.intercept_
```

[163]:
```
(array([     0.        , 180664.33566432, -48548.95104895,   4120.04662005]),
 -121333.33333330264)
```

[164]:
```
df.plot.scatter('Level', 'Salary', color='blue')
plt.plot(
    np.array(df['Level']),
    model_deg3.coef_[1] * np.array(df['Level'])
        + model_deg3.coef_[2] * np.array(df['Level']) ** 2
        + model_deg3.coef_[3] * np.array(df['Level']) ** 3
        + model_deg3.intercept_,
    color='red'
)
plt.show()
```



- 4 Degree

[165]:
```
quartic = PolynomialFeatures(degree=4)
X_deg4 = quartic.fit_transform(X_train)
X_deg4
```

[165]:
```
array([[1.000e+00, 1.000e+00, 1.000e+00, 1.000e+00, 1.000e+00],
       [1.000e+00, 2.000e+00, 4.000e+00, 8.000e+00, 1.600e+01],
       [1.000e+00, 3.000e+00, 9.000e+00, 2.700e+01, 8.100e+01],
       [1.000e+00, 4.000e+00, 1.600e+01, 6.400e+01, 2.560e+02],
       [1.000e+00, 5.000e+00, 2.500e+01, 1.250e+02, 6.250e+02],
       [1.000e+00, 6.000e+00, 3.600e+01, 2.160e+02, 1.296e+03],
       [1.000e+00, 7.000e+00, 4.900e+01, 3.430e+02, 2.401e+03],
       [1.000e+00, 8.000e+00, 6.400e+01, 5.120e+02, 4.096e+03],
       [1.000e+00, 9.000e+00, 8.100e+01, 7.290e+02, 6.561e+03],
       [1.000e+00, 1.000e+01, 1.000e+02, 1.000e+03, 1.000e+04]])
```

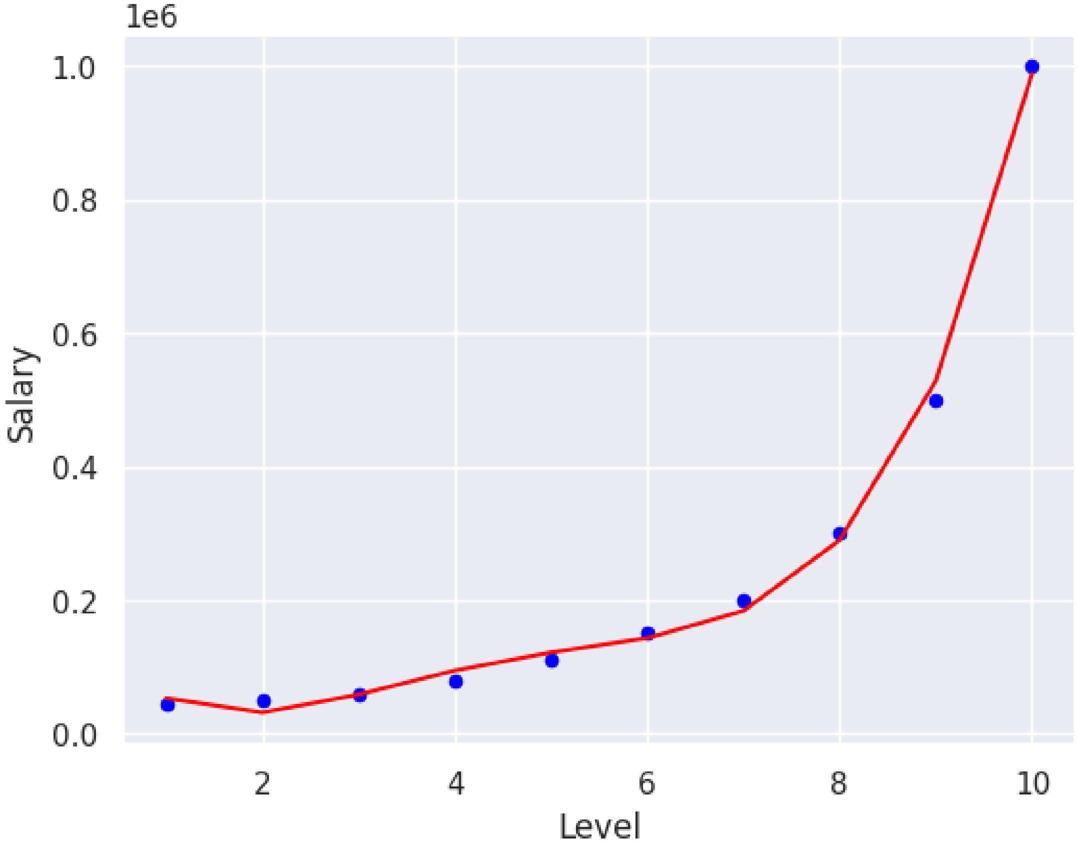
[166]:
```
model_deg4 = LinearRegression()
model_deg4.fit(X_deg4, y_train)
```

```
[166]: LinearRegression()
```

```
[167]: model_deg4.coef_, model_deg4.intercept_
```

```
[167]: (array([      0.        , -211002.33100292,   94765.44289063,
          -15463.28671331,     890.15151515]),
        184166.66666719783)
```

```
[168]: df.plot.scatter('Level', 'Salary', color='blue')
       plt.plot(
           np.array(df['Level']),
           model_deg4.coef_[1] * np.array(df['Level'])
               + model_deg4.coef_[2] * np.array(df['Level']) ** 2
               + model_deg4.coef_[3] * np.array(df['Level']) ** 3
               + model_deg4.coef_[4] * np.array(df['Level']) ** 4
               + model_deg4.intercept_,
           color='red'
       )
       plt.show()
```



4 Linear Regression with Gradient Descent

[107]:

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(style='darkgrid')
```

[108]:

```python
df = pd.read_csv('https://github.com/codebasics/py/raw/master/ML/1_linear_reg/
 ↪homeprices.csv')
df.head()
```

[108]:

```
   area   price
0  2600  550000
1  3000  565000
2  3200  610000
3  3600  680000
4  4000  725000
```

[109]:

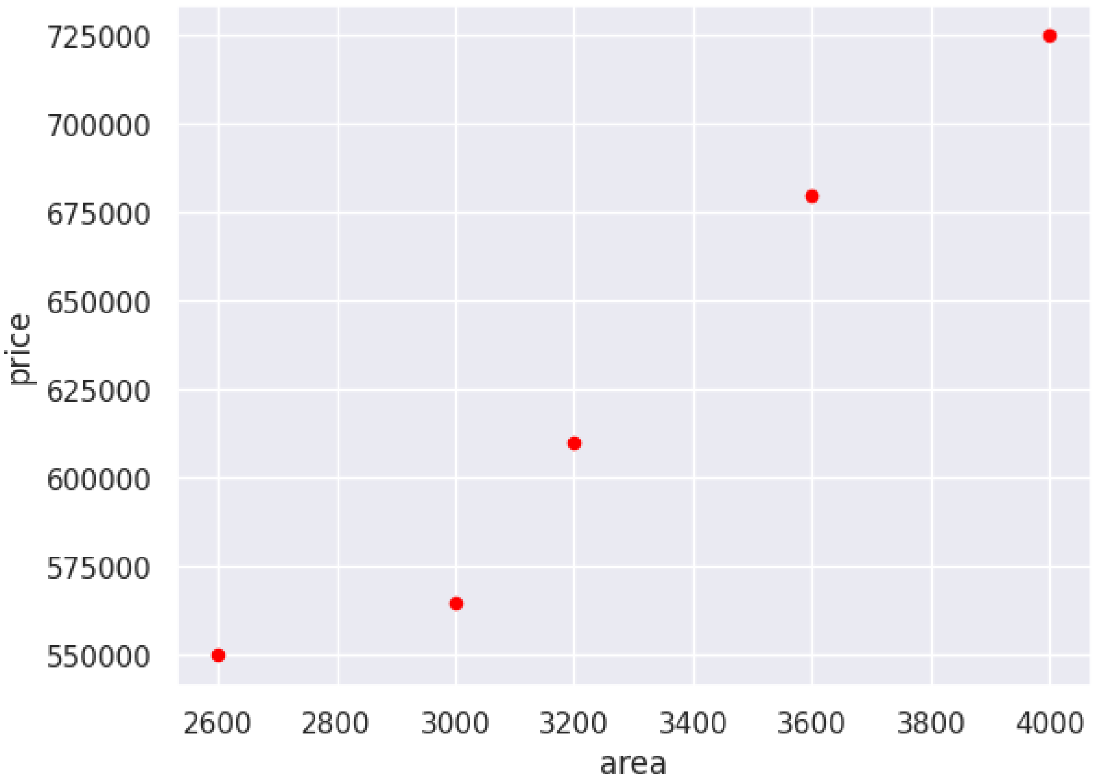
```python
df.describe()
```

[109]:

```
              area          price
count     5.000000       5.000000
mean   3280.000000  626000.000000
std     540.370243   74949.983322
min    2600.000000  550000.000000
25%    3000.000000  565000.000000
50%    3200.000000  610000.000000
75%    3600.000000  680000.000000
max    4000.000000  725000.000000
```

[110]:

```python
df.plot.scatter('area', 'price', color='red')
plt.show()
```



[111]:

```
X_train = df.drop('price', axis=1)
y_train = df.drop('area', axis=1)
```

[112]:

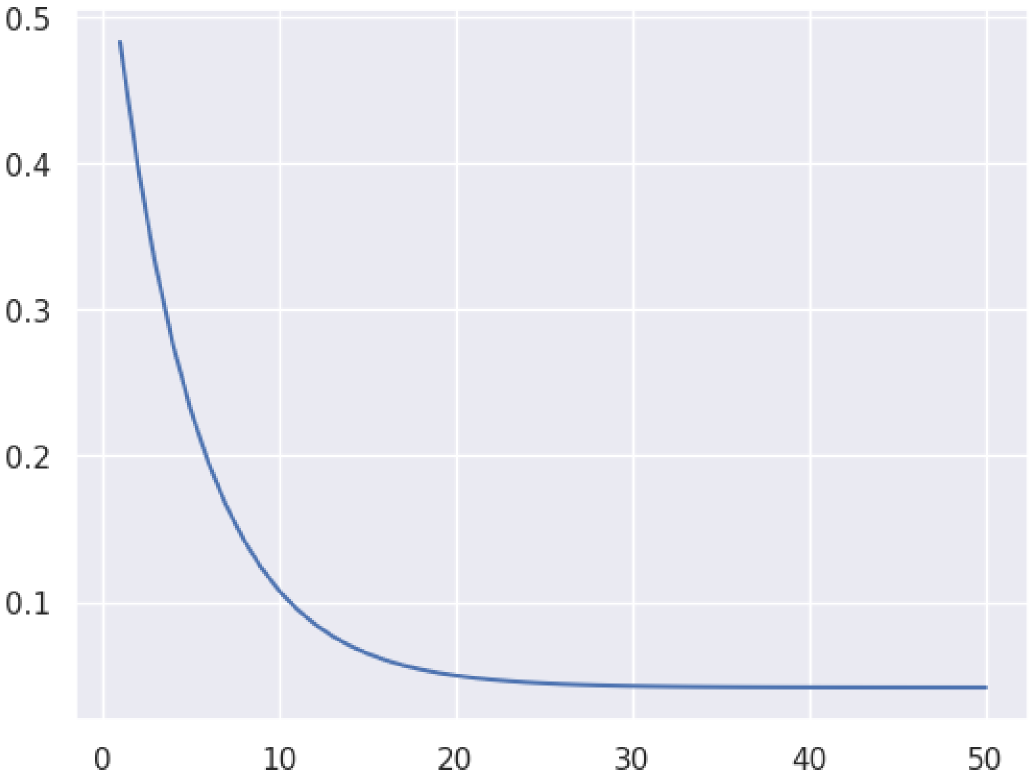
```
from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
y_train_scaled = scaler.fit_transform(y_train)
```

[113]:

```
from sklearn.metrics import mean_squared_error

def gradient_descent(X, y, theta, lr=0.01, steps=100):
    N = len(y)
    cost_history = np.zeros(steps)
    theta_history = np.zeros((steps, 2))
    for i in range(steps):
        prediction = np.dot(X, theta)
        theta = theta - (1 / N) * lr * (X.T.dot((prediction - y)))
        theta_history[i, :] = theta.T
        cost_history[i] = mean_squared_error(prediction, y)

    return theta, cost_history, theta_history
```

[114]:
```
lr = 0.1
it = 50
theta = np.random.randn(2, 1)

X_b = np.c_[
    X_train_scaled,
    np.ones((len(X_train_scaled), 1))
]

_, cost, theta = gradient_descent(X_b, y_train_scaled, theta, lr, it)
```

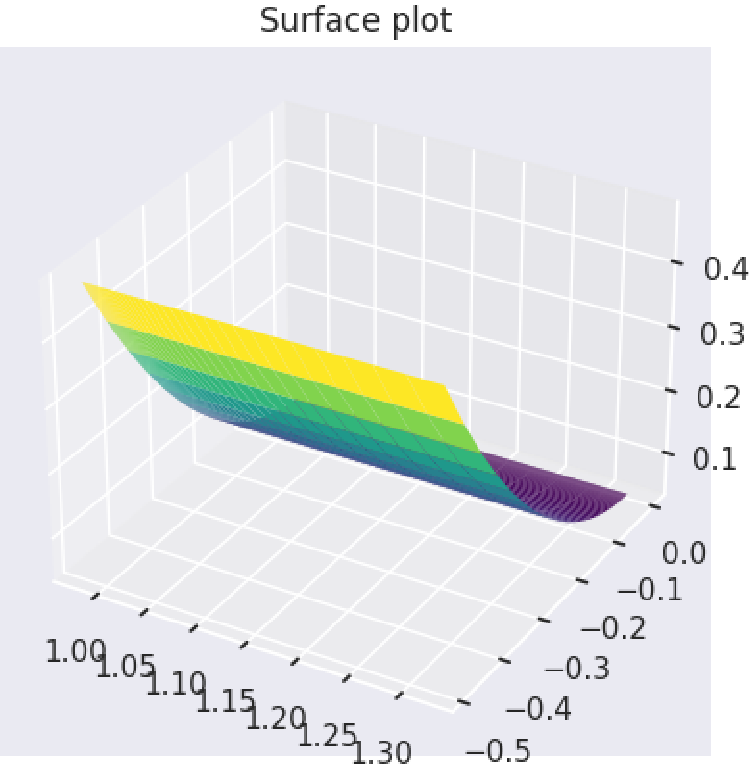
[115]:
```
sns.lineplot(x=range(1, 51), y=cost)
plt.show()
```



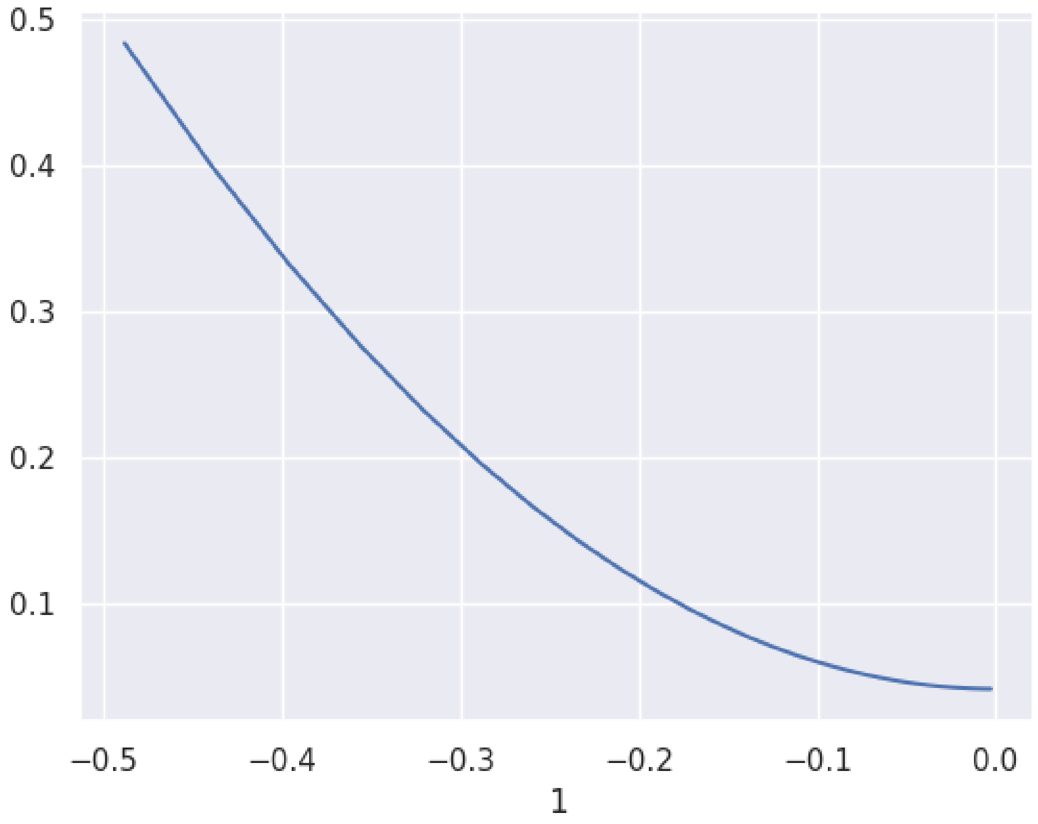
[116]:
```
np.array([1,2])
theta_df = pd.DataFrame(theta)
```

[117]:
```
fig = plt.figure()
ax = plt.axes(projection='3d')

ax.plot_surface(theta_df[0], theta_df.drop(0, axis=1), np.array(cost).
 ↪reshape(-1, 1), cmap='viridis', edgecolor='none')
```

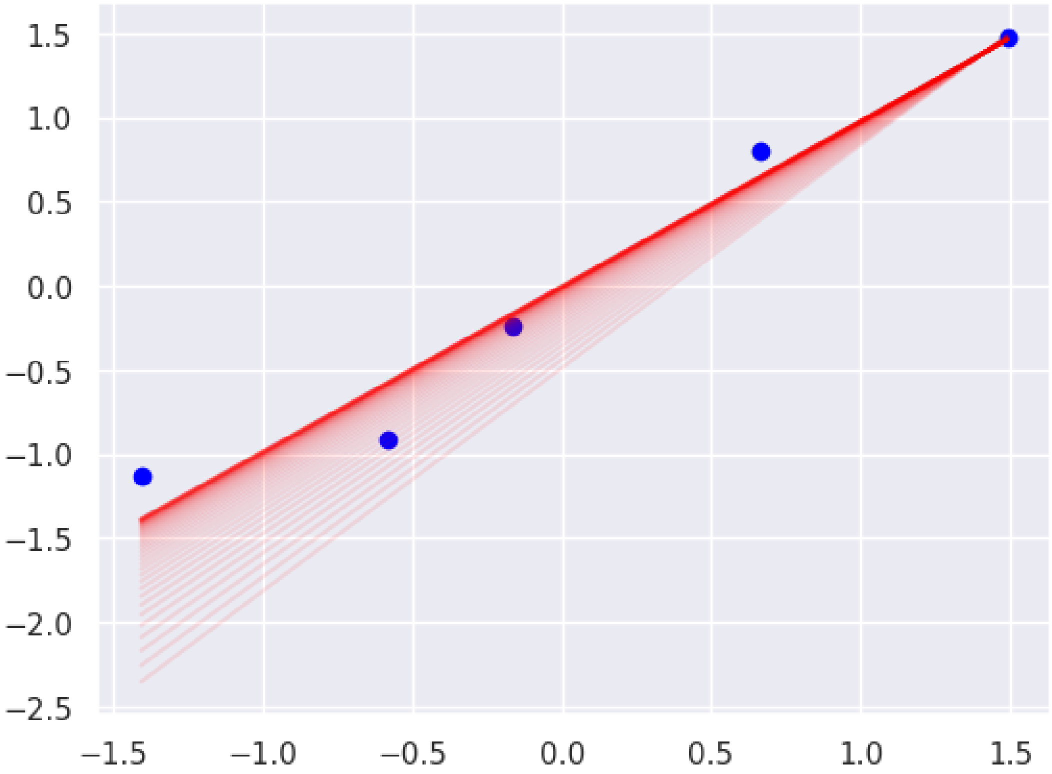
```
ax.set_title('Surface plot')
plt.show()
```



[118]:
```
sns.lineplot(x=theta_df[1], y=cost)
plt.show()
```



[119]:
```
plt.scatter(X_train_scaled, y_train_scaled, color='blue')
for t in theta:
    plt.plot(
        X_train_scaled,
        t[0] * X_train_scaled + t[1],
        color='red',
        alpha=0.1
    )
plt.show()
```

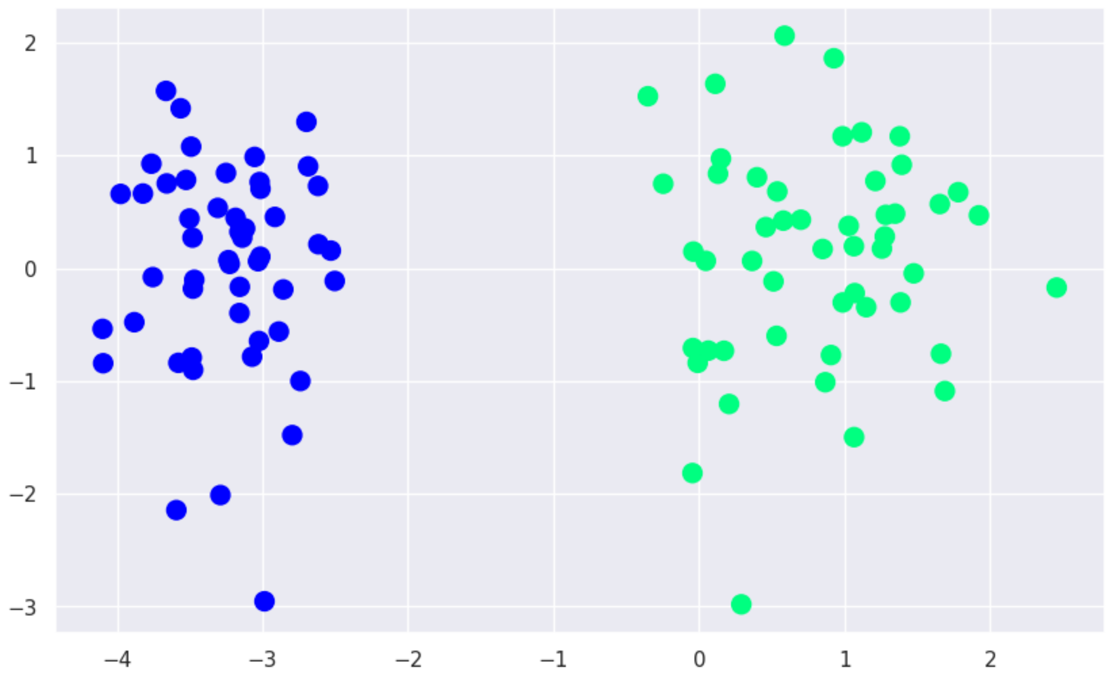


5 Logistic Regression

[120]:
```
from sklearn.datasets import make_classification
import numpy as np
X, y = make_classification(n_samples=100, n_features=2,␣
↪n_informative=1,n_redundant=0,n_classes=2, n_clusters_per_class=1,␣
↪random_state=41,hypercube=False,class_sep=20)
```

[123]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(10,6))
plt.scatter(X[:,0],X[:,1],c=y,cmap='winter',s=100)
```

[123]:
```
<matplotlib.collections.PathCollection at 0x7f46626c2ee0>
```



[124]:

```
from sklearn.linear_model import LogisticRegression
lor = LogisticRegression(penalty='none',solver='sag')
lor.fit(X,y)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_logistic.py:1173:
FutureWarning:

`penalty='none'`has been deprecated in 1.2 and will be removed in 1.4. To keep
the past behaviour, set `penalty=None`.

/usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning:

The max_iter was reached which means the coef_ did not converge
```

[124]:

```
LogisticRegression(penalty='none', solver='sag')
```

[125]:

```
print(lor.coef_)
print(lor.intercept_)
```

```
[[4.79291636 0.20533276]]
[5.76753249]
```

[126]:

```
m1 = -(lor.coef_[0][0]/lor.coef_[0][1])
b1 = -(lor.intercept_/lor.coef_[0][1])
```

[127]:

```
x_input = np.linspace(-3,3,100)
y_input = m1*x_input + b1
```

[129]:

```
def sigmoid(z):
    return 1/(1 + np.exp(-z))
```

[128]:

```
def gd(X,y):

    X = np.insert(X,0,1,axis=1)
    weights = np.ones(X.shape[1])
    lr = 0.5

    for i in range(5000):
        y_hat = sigmoid(np.dot(X,weights))
        weights = weights + lr*(np.dot((y-y_hat),X)/X.shape[0])

    return weights[1:],weights[0]
```

[130]:

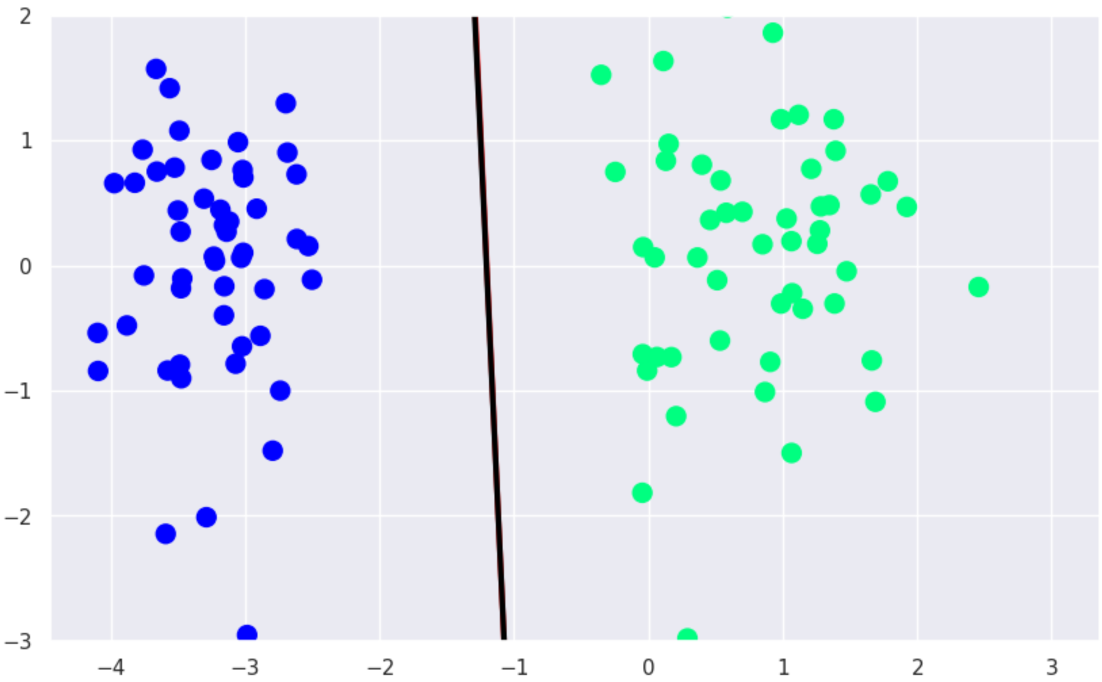
```
coef_,intercept_ = gd(X,y)
m = -(coef_[0]/coef_[1])
b = -(intercept_/coef_[1])
x_input1 = np.linspace(-3,3,100)
y_input1 = m*x_input1 + b
```

[131]:

```
plt.figure(figsize=(10,6))
plt.plot(x_input,y_input,color='red',linewidth=3)
plt.plot(x_input1,y_input1,color='black',linewidth=3)
plt.scatter(X[:,0],X[:,1],c=y,cmap='winter',s=100)
plt.ylim(-3,2)
```

[131]:

```
(-3.0, 2.0)
```


2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3