Lecture 4: Linear Models

Linear Regression

Linear regression models the relationship between an **independent (explanatory)** variable X and a quantitative **dependent (response)** value Y.

Examples:

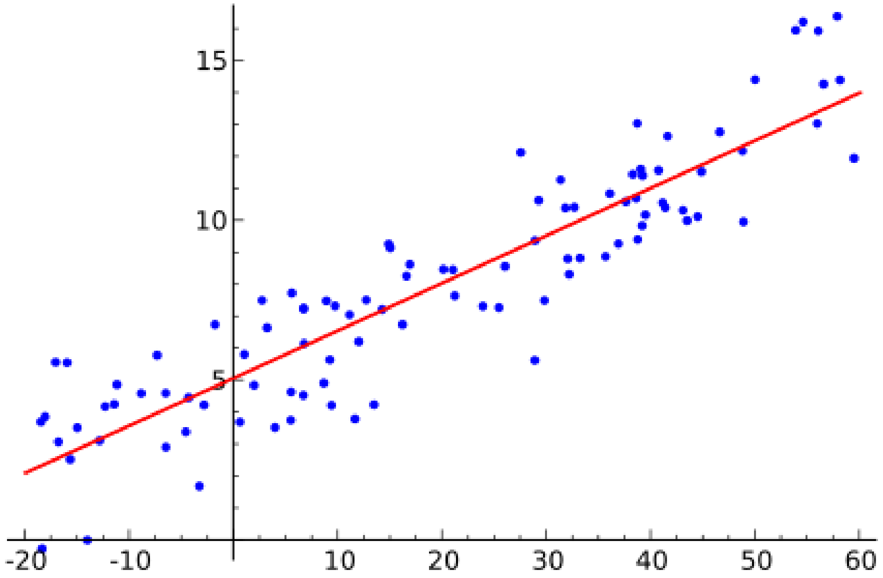
explanatory variable, X	dependent variable, Y
square footage	house price
advertising dollars spent	profit
stress	lifespan
?	?

Simple Linear Regression (SLR)

Data: We have n samples $(x_1, y_1),\ (x_2, y_2), \ldots, (x_n, y_n)$.

Model: $y \sim \beta_0 + \beta_1 x$

Goal: Find the best values of β_0 and β_1, denoted $\hat{\beta}_0$ and $\hat{\beta}_1$, so that the prediction $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$ "best fits" the data.



Theorem.

The best parameters in the *least squares sense* are:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

$$\text{and} \qquad \hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x}.$$

where $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and $\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$.

Simple linear regression with python

Python packages for regression

There are several different python packages that do regression:

1. statsmodels
2. scikit-learn
3. SciPy
4. ...

Today, we'll look at both `statsmodels` and `scikit-learn`. One can also use SciPy for linear regression, but its built-in functionality is comparatively limited.

Example dataset

To illustrate linear regression, we'll use the 'Advertising' dataset from here

For 200 different 'markets' (think different cities), this dataset consists of the number of sales of a particular product as well as the advertising budget for different media: TV, radio, and newspaper.

We’ll use linear regression to study the effect of advertising on sales.
Here, sales is the dependent variable and the budgets are the independent variables. This might help inform or evaluate an advertising strategy for this product.

imports and setup

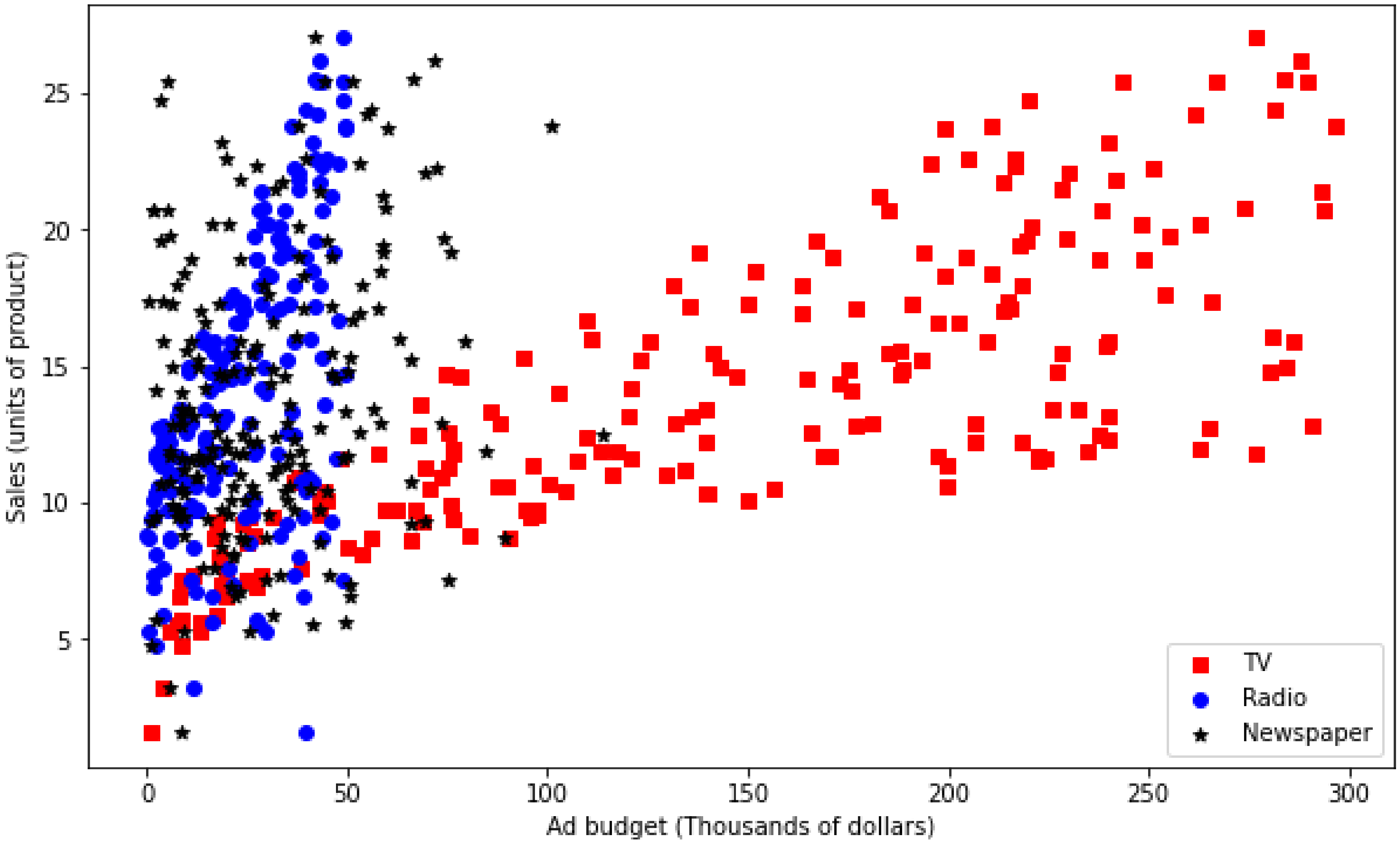
```python
import scipy as sc
from scipy.stats import norm

import pandas as pd
import numpy as np
import statsmodels.formula.api as sm
from sklearn import linear_model

import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = (10, 6)

from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
```

```python
advert = pd.read_csv('Advertising.csv',index_col=0)
advert
```

	TV	Radio	Newspaper	Sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
...	...	...	...	...
197	94.2	4.9	8.1	9.7
198	177.0	9.3	6.4	12.8
199	283.6	42.0	66.2	25.5
200	232.1	8.6	8.7	13.4

Plot and describe the data

```
plt.scatter(x=advert['TV'],y=advert['Sales'],c='r',marker='s',label='TV')
plt.scatter(x=advert['Radio'],y=advert['Sales'],c='b',marker='o',label='Radio')
plt.scatter(x=advert['Newspaper'],y=advert['Sales'],c='k',marker='*',label='Newspaper')

plt.legend(loc=4)
plt.xlabel('Ad budget (Thousands of dollars)')
plt.ylabel('Sales (units of product)')
plt.show()
```



Sales (units of product)
25
20
15
10
5
0
50
100
150
200
250
300
Ad budget (Thousands of dollars)
TV
Radio
Newspaper

```
advert.describe()
```

	TV	Radio	Newspaper	Sales
count	200	200	200	200
mean	147.042	23.264	30.554	14.0225
std	85.8542	14.8468	21.7786	5.21746
min	0.7	0	0.3	1.6
25%	74.375	9.975	12.75	10.375
50%	149.75	22.9	25.75	12.9
75%	218.825	36.525	45.1	17.4
max	296.4	49.6	114	27

Observations

1. From the plot, it is clear that there is a relationship between the advertising budgets and sales. Basically, the more money spent, the larger the number of sales.

- The most money was spent on TV advertising. The amount for Radio and Newspaper is about the same in all markets, whereas the standard deviation for TV advertising is larger.

Questions

1. How can we quantify the relationship between advertising and sales? Can we predict the effect of each ad media on sales? Is the relationship linear?

- Which of the different ad media (TV, Radio, Newspaper) are the most effective at generating sales?
- Are there interactions between the different ad media?

First, let's just look at the **effect of TV advertising on sales**. We use the linear regression model

$$Sales = \beta_0 + \beta_1 * TV.$$

```python
ad_TV_ols = sm.ols(formula="Sales ~ TV", data=advert).fit()
ad_TV_ols.summary()
```

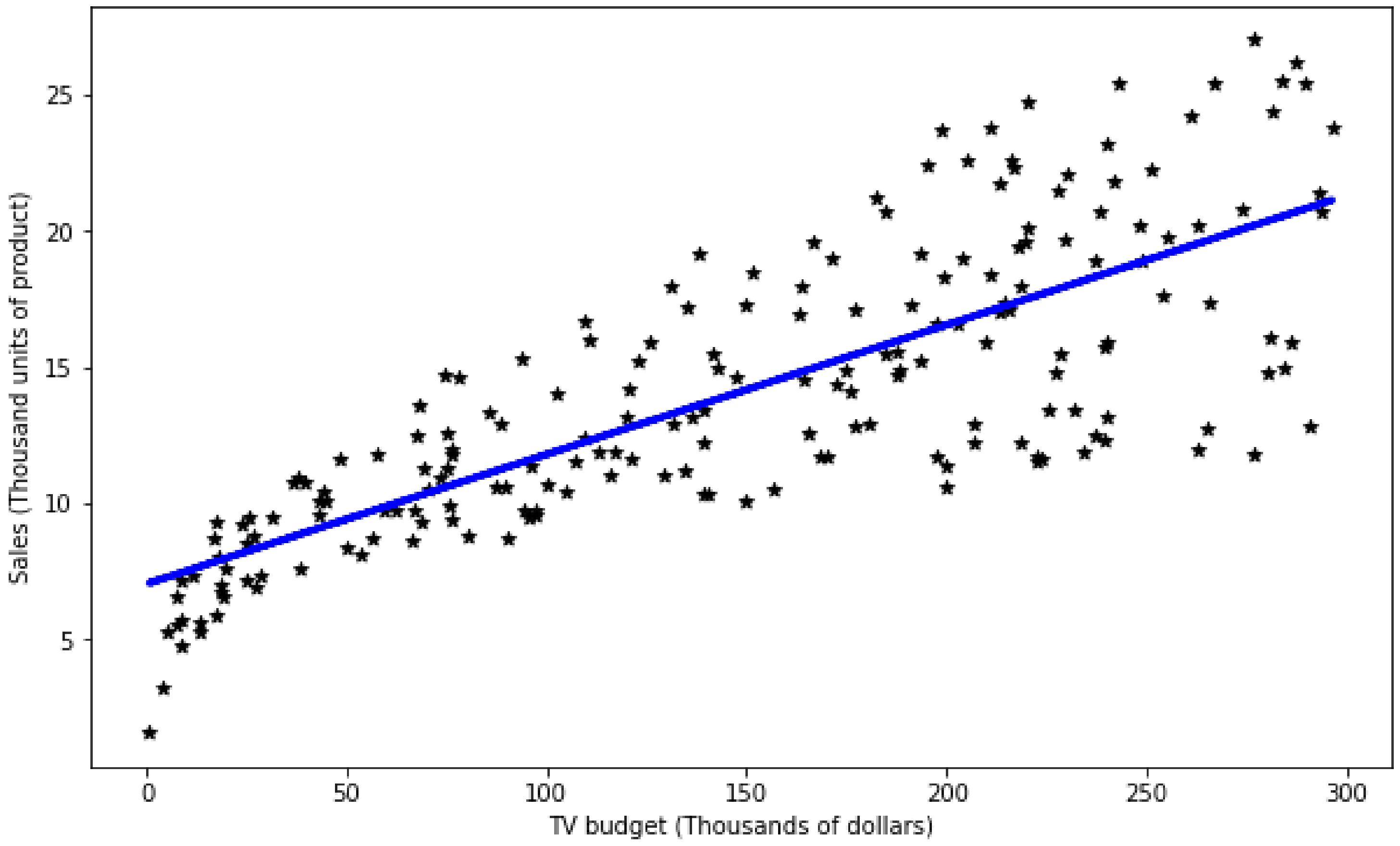
OLS Regression Results

Dep. Variable:	Sales	R-squared:	0.612
Model:	OLS	Adj. R-squared:	0.610
Method:	Least Squares	F-statistic:	312.1
Date:	Wed, 30 Mar 2022	Prob (F-statistic):	1.47e-42
Time:	19:42:25	Log-Likelihood:	-519.05
No. Observations:	200	AIC:	1042.
Df Residuals:	198	BIC:	1049.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	7.0326	0.458	15.360	0.000	6.130	7.935
TV	0.0475	0.003	17.668	0.000	0.042	0.053

Omnibus:	0.531	Durbin-Watson:	1.935
Prob(Omnibus):	0.767	Jarque-Bera (JB):	0.669
Skew:	-0.089	Prob(JB):	0.716
Kurtosis:	2.779	Cond. No.	338.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.



Sales (Thousand units of product)
25
20
15
10
5
0
50
100
150
200
250
300
TV budget (Thousands of dollars)

Interpretation and discussion

The intercept of the line is $\hat{\beta}_0 = 7.032$. This means that without any TV advertising, the model predicts that 7,032 units of product will be sold.

The slope of the line is $\hat{\beta}_1 = 0.0475$. This means that the model predicts that for every additional $1k spent on TV advertising, an additional 47.5 units of product are sold.

So how good is this fit?

One way to measure the quality of the fit is to look at the sum of the squared residuals,

$$SSR = \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2.$$

SSR is the quantity that we minimized to find $\hat{\beta}_0$ and $\hat{\beta}_1$ in the first place (We called it $J(\beta_0, \beta_1)$.) If this number is very small, then the model fits the data very well.

```
ad_TV_ols.ssr
```

2102.5305831313512

But how small is small?

This number, SSR, is difficult to interpret by itself.
A more easily interpretable number is the R^2 **value**.

The R^2 value is an alternative way to measure how good of a fit the model is to the data. The benefit of the R^2 value over the SSR is that it is a proportion (takes values between 0 and 1) so it is easier to interpret what a *good* value is. We first define the sum of squared residuals (SSR) and total sum of squares (TSS) by

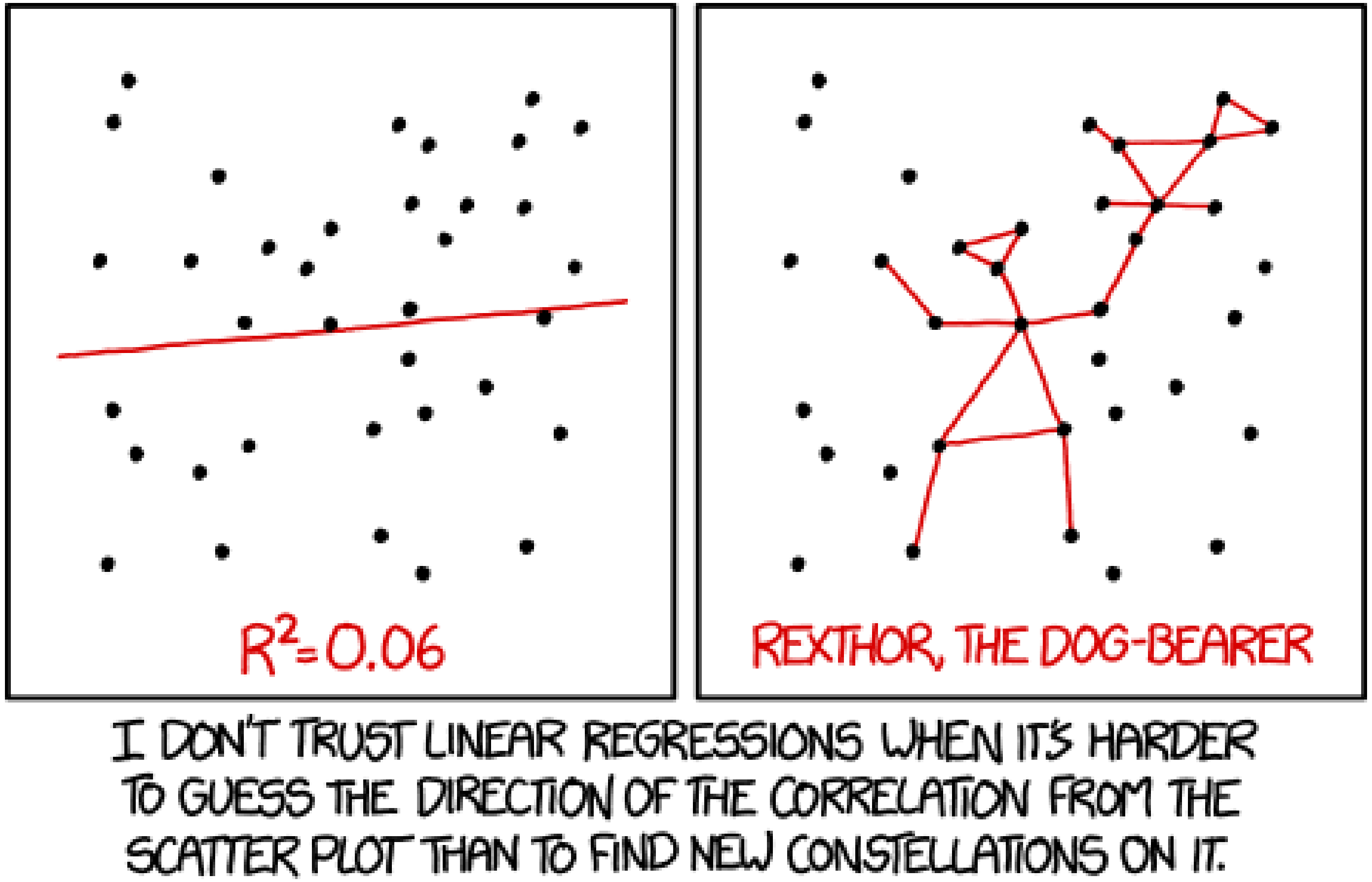
$$SSR = \sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 \qquad \text{and} \qquad TSS = \sum_{i=1}^{n}(y_i - \bar{y})^2.$$

SSR measures the amount of variability left unexplained after the linear regression. TSS measures the total variance in the dependent variable. We compute the R^2 value as

$$R^2 = 1 - \frac{SSR}{TSS}.$$

This is the **proportion of the variance explained by the model**. A model is good if the R^2 value is nearly one (the model explains all of the variance in the data).

In our model, the value is $R^2 = 0.612$, which isn't bad. The model explains 61% of the variability in sales.



Repeating the simple linear regression with scikit-learn

```
lr = linear_model.LinearRegression() # create a linear regression object

# scikit-learn doesn't work as well with pandas, so we have to extract values
x = advert['TV'].values.reshape(advert['TV'].shape[0],1)
y = advert['Sales'].values.reshape(advert['Sales'].shape[0],1)

lr.fit(X=x, y=y)

plt.scatter(x, y,  color='black')
plt.plot(x, lr.predict(x), color='blue', linewidth=3)

plt.xlabel('TV budget (Thousands of dollars)')
plt.ylabel('Sales (Thousand units of product)')
plt.show()
```

Hypothesis testing in linear regression

In descriptive statistics, one seeks just to describe the data. In the present setting, we have described how the response variable linearly depends on the predictor variable by minimizing the residual sum of squares (RSS).

The statistical inference way of looking at this problem would be to suppose that there exists a ground truth population with x and y related by

$$y = \beta_0 + \beta_1 x$$

for some unknown values of β_0 and β_1.

Our sampled data consists of points (x_i, y_i) of the form

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i.$$

Here ε_i are random variable (say normally distributed) that we think of as "error" being introduced into the samples. The job of the statistician is to *infer* the values of β_0 and β_1 from the noisy data.

This is precisely the setting we were in when determining whether a coin was fair. There, we had a sample proportion of heads (analogous to the samples (x_i, y_i) here.) We used the Central Limit Theorem to say that the variance (standard error) is given by

$$SE(\hat{\mu})^2 = \sigma^2/n$$

Using this value and assuming the null hypothesis (the coin is fair), we could evaluate the likelihood of obtaining a sample as extreme as the one obtained.

In simple linear regression, we will take the null hypothesis to be

$$H_0 : \text{There is no linear relationship between } x \text{ and } y \iff \beta_1 = 0$$

with alternative

$$H_a : \text{There is a linear relationship between } x \text{ and } y \iff \beta_1 \neq 0$$

We assume that ε is a normal random variable with zero mean and variance σ^2. Using similar ideas as above, the standard error for $\hat{\beta}_0$ and $\hat{\beta}_1$ (estimates of true parameters in this model) are computed to be

$$SE(\hat{\beta}_0)^2 = \sigma^2 \left(\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \right) \quad \text{and} \quad SE(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

For this hypothesis test, the test statistic is

$$t = \frac{\hat{\beta}_1 - 0}{SE(\hat{\beta}_1)},$$

which under the assumptions of the null hypothesis, is distributed according to the t distribution with $n-2$ degrees of freedom. The p-value is computed as the probability of observing a value as extreme as $|t|$. A small p-value is interpreted to mean that there is an association between the independent and dependent variables.

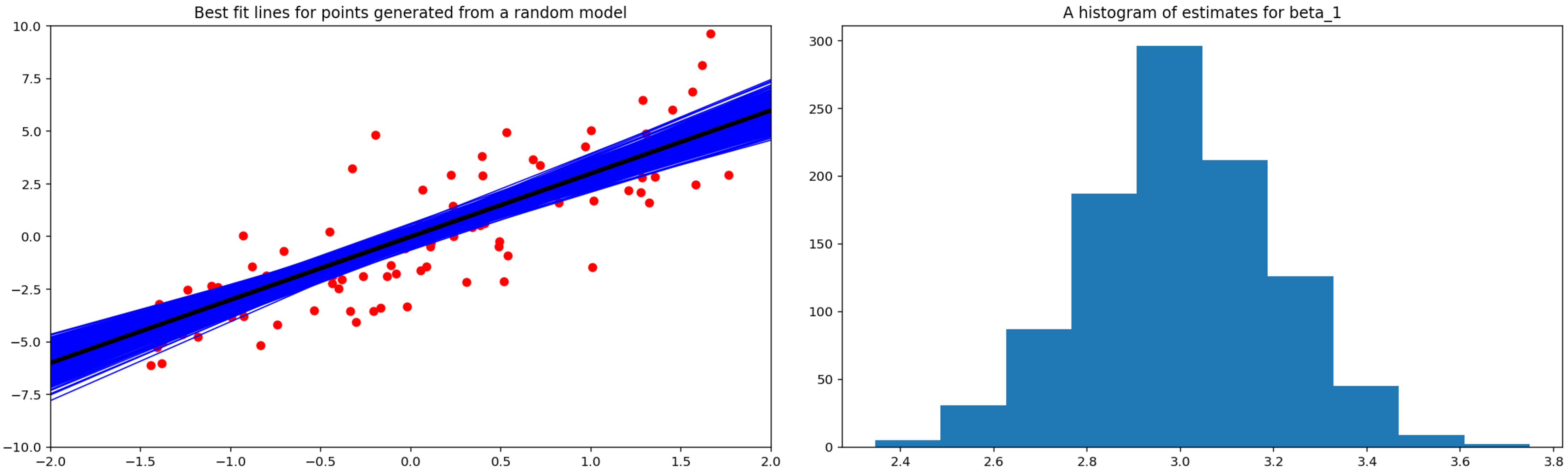
An experiment

Before we go back and discuss the p-values for the advertising data, let's look at some synthetic data. We generate 100 random points according to the model

$$y = 3 * x + \varepsilon,$$

where ε is normally distributed with mean zero and standard deviation 2. The best fit is found and this process is repeated 1000 times.

```
f= lambda x: 3*x
x = np.linspace(-2,2,3).reshape(3,1) #Define 3 values which will be used to plot regression line
plt.figure(0)

sample_size = 100
betaOnes = []
for ii in range(1000):
    xp = norm.rvs(size=sample_size)
    yp = f(xp)+norm.rvs(size=sample_size,scale=2)

    if ii == 0: plt.plot(xp,yp,'ro')

    lr = linear_model.LinearRegression()
    lr.fit(X=xp.reshape(100,1), y=yp)
    plt.plot(x,lr.predict(x),color='blue',linewidth=1)

    # Collect the slopes
    betaOnes.append(lr.coef_[0])

plt.plot(xp,f(xp),'k',linewidth=3)
plt.xlim(-2,2)
plt.ylim(-10,10)
plt.title('Best fit lines for points generated from a random model')
plt.show()

plt.figure(1)
plt.hist(betaOnes)
plt.title('A histogram of estimates for beta_1')
plt.show()
```


Best fit lines for points generated from a random model
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
A histogram of estimates for beta_1
300
250
200
150
100
50
0
2.4
2.6
2.8
3.0
3.2
3.4
3.6
3.8

Now, let's return to the linear regression model of sales on TV advertising,

$$Sales = \beta_0 + \beta_1 * TV \,.$$

Looking at the `statsmodels` output, we see that the p-values for both the intercept and the slope are very small. The probability of obtaining these values is nearly zero, assuming H_0 is true. So, we reject the null hypothesis and say there is a linear association between TV advertising (independent variable) and sales (dependent variable).

Other advertisement methods?

```
ad_Radio_ols = sm.ols(formula="Sales ~ Radio", data=advert).fit()
ad_Radio_ols.summary()
```

OLS Regression Results

Dep. Variable:	Sales	R-squared:	0.332
Model:	OLS	Adj. R-squared:	0.329
Method:	Least Squares	F-statistic:	98.42
Date:	Wed, 06 Apr 2022	Prob (F-statistic):	4.35e-19
Time:	16:41:34	Log-Likelihood:	-573.34
No. Observations:	200	AIC:	1151.
Df Residuals:	198	BIC:	1157.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	9.3116	0.563	16.542	0.000	8.202	10.422
Radio	0.2025	0.020	9.921	0.000	0.162	0.243

Omnibus:	19.358	Durbin-Watson:	1.946
Prob(Omnibus):	0.000	Jarque-Bera (JB):	21.910
Skew:	-0.764	Prob(JB):	1.75e-05
Kurtosis:	3.544	Cond. No.	51.4

```
ad_Newspaper_ols = sm.ols(formula="Sales ~ Newspaper", data=advert).fit()
ad_Newspaper_ols.summary()
```

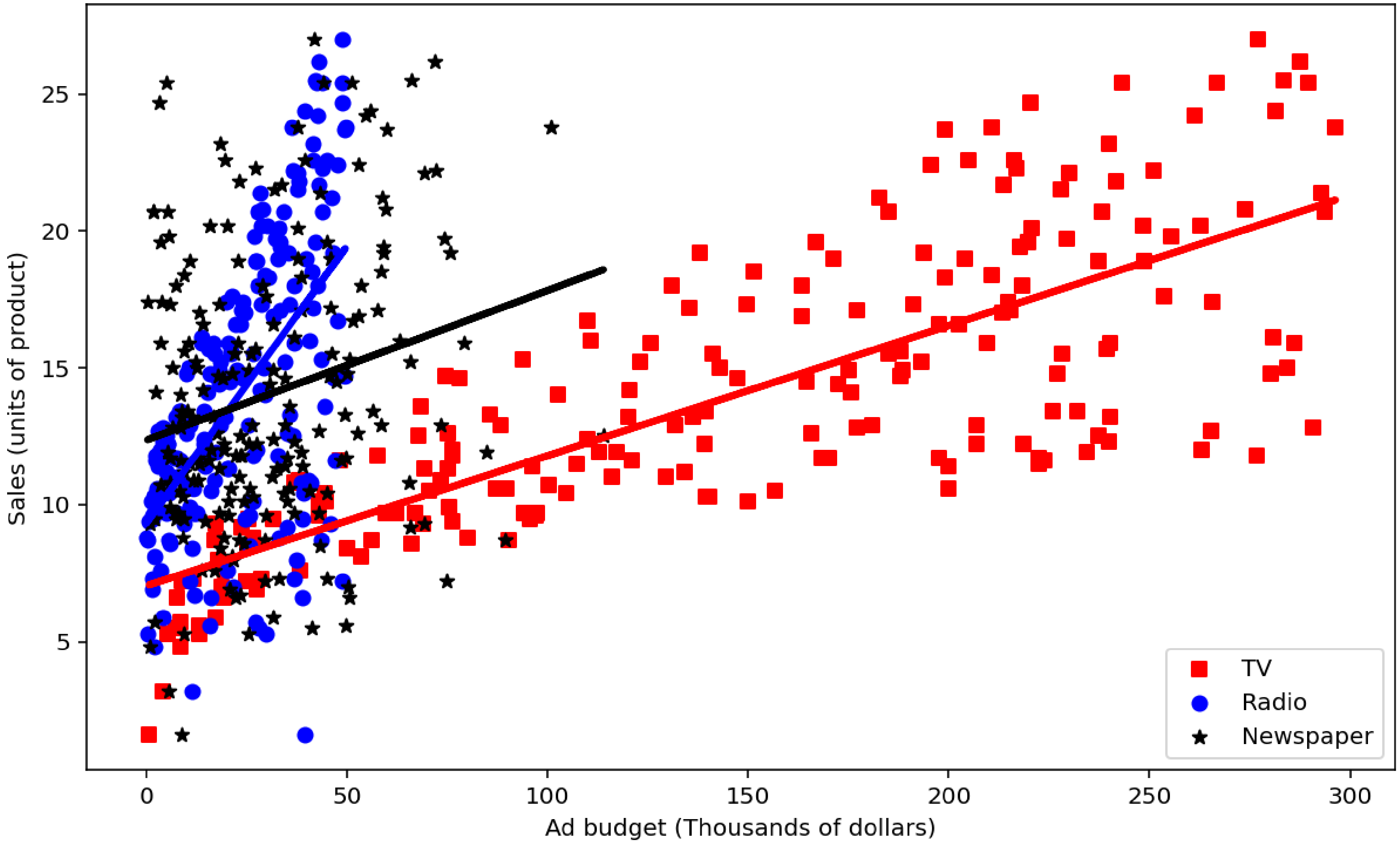
OLS Regression Results

Dep. Variable:	Sales	R-squared:	0.052
Model:	OLS	Adj. R-squared:	0.047
Method:	Least Squares	F-statistic:	10.89
Date:	Wed, 06 Apr 2022	Prob (F-statistic):	0.00115
Time:	16:43:18	Log-Likelihood:	-608.34
No. Observations:	200	AIC:	1221.
Df Residuals:	198	BIC:	1227.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	12.3514	0.621	19.876	0.000	11.126	13.577
Newspaper	0.0547	0.017	3.300	0.001	0.022	0.087

Omnibus:	6.231	Durbin-Watson:	1.983
Prob(Omnibus):	0.044	Jarque-Bera (JB):	5.483
Skew:	0.330	Prob(JB):	0.0645
Kurtosis:	2.527	Cond. No.	64.7

```python
plt.scatter(x=advert['TV'],y=advert['Sales'],c='r',marker='s',label='TV')
plt.scatter(x=advert['Radio'],y=advert['Sales'],c='b',marker='o',label='Radio')
plt.scatter(x=advert['Newspaper'],y=advert['Sales'],c='k',marker='*',label='Newspaper')
plt.legend(loc=4)

plt.plot(advert['TV'],ad_TV_ols.predict(),c='r',linewidth=3)
plt.plot(advert['Radio'],ad_Radio_ols.predict(),c='b',linewidth=3)
plt.plot(advert['Newspaper'],ad_Newspaper_ols.predict(),c='k',linewidth=3)

plt.xlabel('Ad budget (Thousands of dollars)')
plt.ylabel('Sales (units of product)')
plt.show()
```


Sales (units of product)
25
20
15
10
5
0
50
100
150
200
250
300
Ad budget (Thousands of dollars)
TV
Radio
Newspaper

Interpretation

So what is the most effective advertising media?

The slope for radio is largest, so you might argue that this is the most effective advertising media. For every additional $1k spent on Radio advertising, an additional 202 units of product are sold. (Compare to 54.7 for newspaper and 47.5 for TV.)

On the other hand, the R^2 value for radio is just 33%. So the model isn't explaining as much of the data as the model for TV advertising ($R^2 = 61\%$), but is explaining more than the model for newspaper advertising ($R^2 = 5\%$).

The main problem with the approach here is that for each advertising media we look at, we're ignoring the ads in the other media. For example, in the model for TV advertising,

$$Sales = \beta_0 + \beta_1 * TV,$$

we're ignoring both Radio and Newspaper advertising.

We need to take all three into account at once. Maybe we can construct a model that looks like

$$Sales = \beta_0 + \beta_1 * TV + \beta_2 * Radio + \beta_3 * Newspaper.$$

This is the idea behind Multiple Linear Regression.

Multiple Linear Regression

Model:

$$Sales = \beta_0 + \beta_1 * TV + \beta_2 * Radio + \beta_3 * Newspaper.$$

```
ad_all_ols = sm.ols(formula="Sales ~ TV + Radio + Newspaper", data=advert).fit()
ad_all_ols.summary()
```

OLS Regression Results

Dep. Variable:	Sales	R-squared:	0.897
Model:	OLS	Adj. R-squared:	0.896
Method:	Least Squares	F-statistic:	570.3
Date:	Wed, 06 Apr 2022	Prob (F-statistic):	1.58e-96
Time:	16:45:18	Log-Likelihood:	-386.18
No. Observations:	200	AIC:	780.4
Df Residuals:	196	BIC:	793.6
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.9389	0.312	9.422	0.000	2.324	3.554
TV	0.0458	0.001	32.809	0.000	0.043	0.049
Radio	0.1885	0.009	21.893	0.000	0.172	0.206
Newspaper	-0.0010	0.006	-0.177	0.860	-0.013	0.011

Omnibus:	60.414	Durbin-Watson:	2.084
Prob(Omnibus):	0.000	Jarque-Bera (JB):	151.241
Skew:	-1.327	Prob(JB):	1.44e-33
Kurtosis:	6.332	Cond. No.	454.

Interpretation

Spending an additional $1,000 on radio advertising results in an increase in sales by 189 units. Radio is the most effective method of advertising.

In multilinear regression, the F-test is a way to test the null hypothesis,

$$H_0 = \text{all coefficients are zero.}$$

In this case, we see that the p-value for the F-statistic is vanishingly small - indicating that our model is significant.

Now let's consider the individual coefficients in the model and their p-values. Note that the coefficients for TV and Radio are approximately the same as for simple linear regression. The coefficient for Newspaper changed significantly. Furthermore, note that the p-value is now very large $p = 0.86$. There is not sufficient evidence to reject the null hypothesis that the Newspaper and Sales variables have no relationship.

So then why did the simple linear regression give that there is a relationship between Newspaper and Sales Variables?

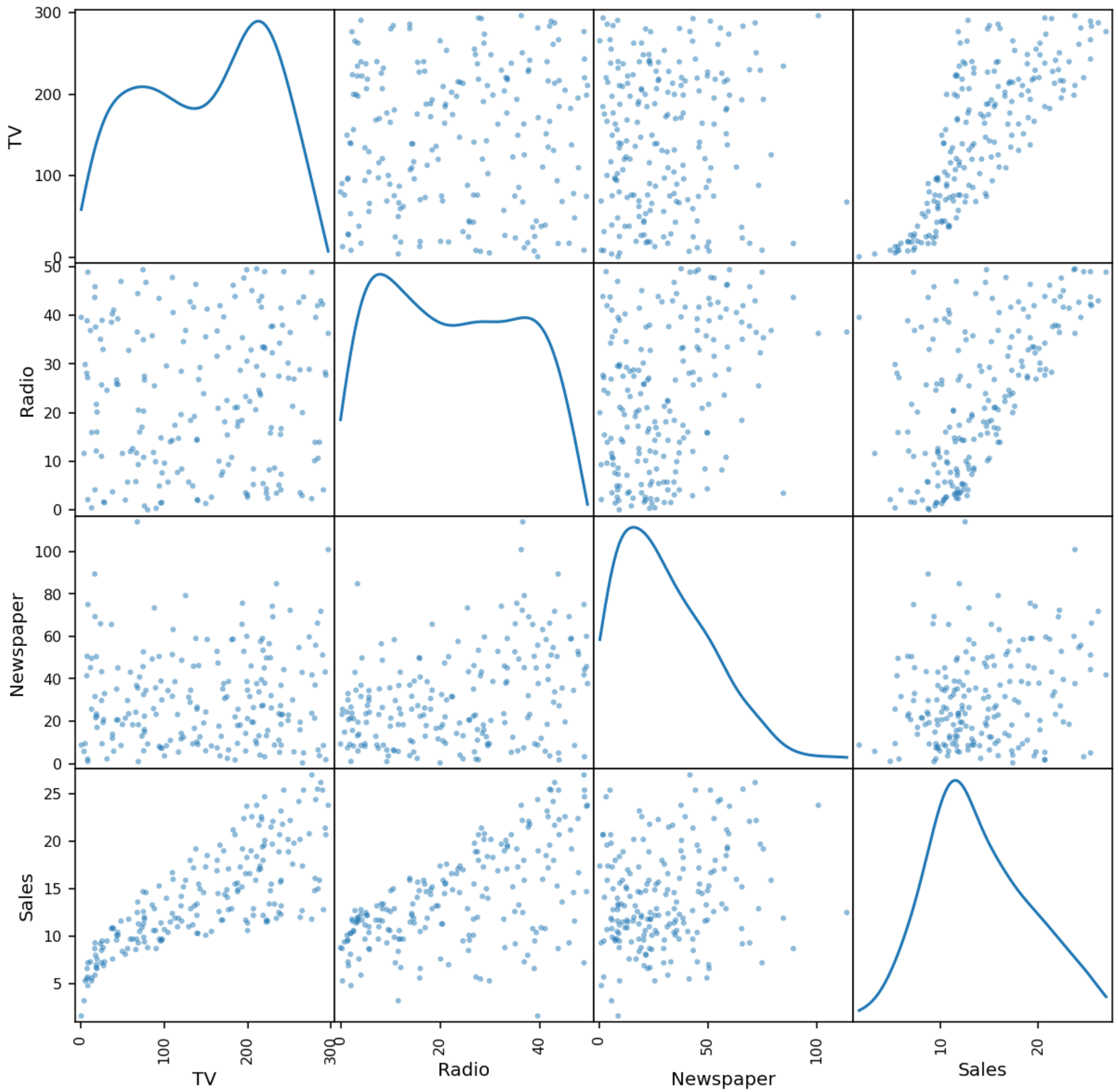
Newspaper is actually a confounder! (Remember the example where temperature is a confounder for pool deaths and ice creams sales.) Let's look at the correlations between the four variables. Recall that correlation between two variables is given by

$$r_{x,y} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{s_x s_y}.$$

Correlation is a number between –1 to +1 and measures how much the two variables vary together.

```
print(advert.corr())
pd.plotting.scatter_matrix(advert, figsize=(10, 10), diagonal='kde')
plt.show()
```

	TV	Newspaper	Newspaper	Sales
TV	1.000000	0.054809	0.056648	0.782224
Radio	0.054809	1.000000	0.354104	0.576223
Newspaper	0.056648	0.354104	1.000000	0.228299
Sales	0.782224	0.576223	0.228299	1.000000


TV
Radio
Newspaper
Sales

The correlation between Newspaper and Radio is 0.35, which implies that in markets where the company advertised using Radio, they also advertised using newspaper. Thus, the influence of Radio on Sales can be incorrectly attributed to Newspaper advertisements!

This leads us to the following linear regression model, where we forget about Newspaper advertisements:

$$\text{Sales} = \beta_0 + \beta_1 * \text{TV budget} + \beta_2 * \text{Radio budget}$$

```
ad_TR_ols = sm.ols(formula="Sales ~ TV + Radio", data=advert).fit()
ad_TR_ols.summary()
```

OLS Regression Results

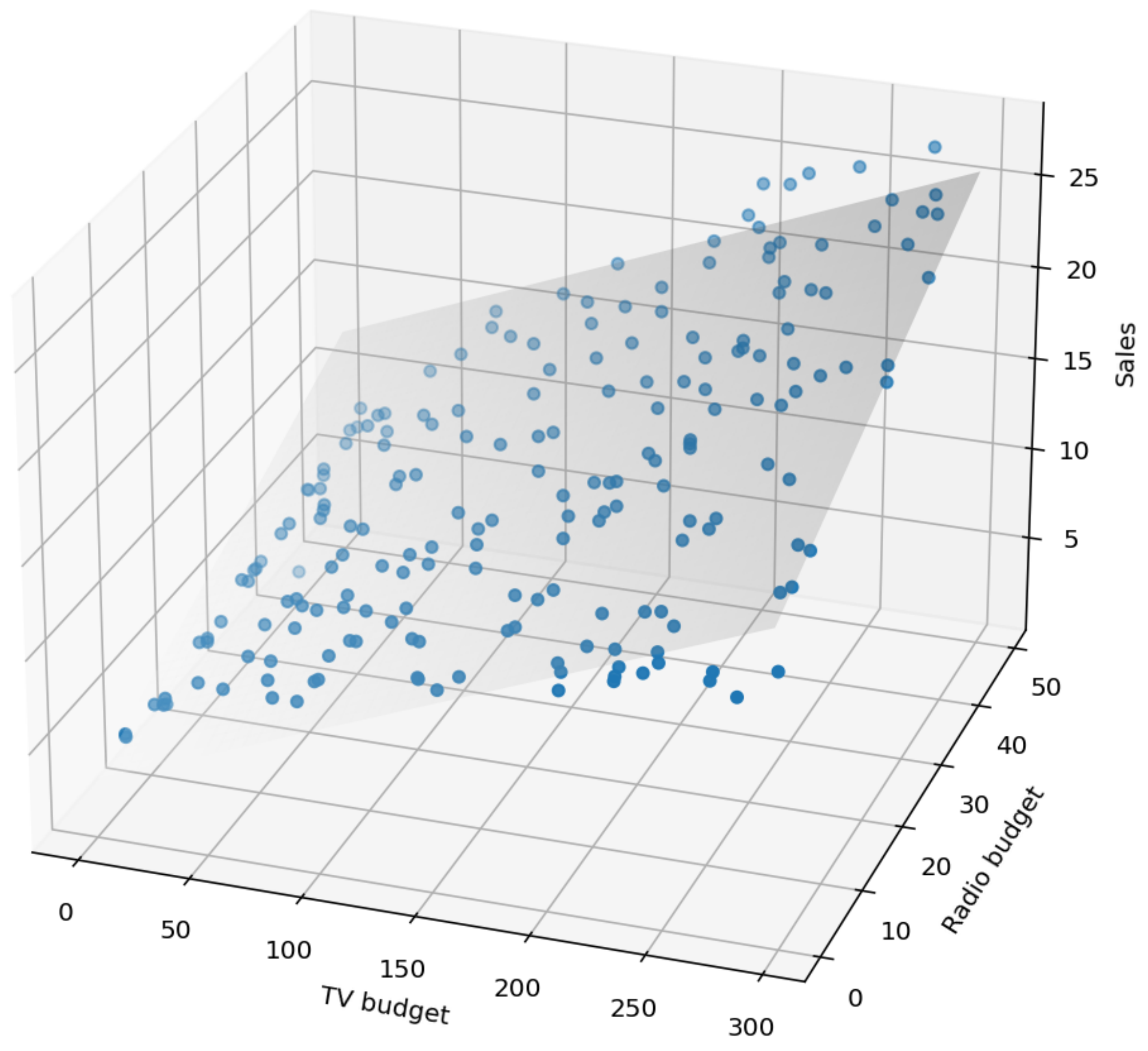
Dep. Variable:	Sales	R-squared:	0.897
Model:	OLS	Adj. R-squared:	0.896
Method:	Least Squares	F-statistic:	859.6
Date:	Wed, 06 Apr 2022	Prob (F-statistic):	4.83e-98
Time:	16:50:24	Log-Likelihood:	-386.20
No. Observations:	200	AIC:	778.4
Df Residuals:	197	BIC:	788.3
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.9211	0.294	9.919	0.000	2.340	3.502
TV	0.0458	0.001	32.909	0.000	0.043	0.048
Radio	0.1880	0.008	23.382	0.000	0.172	0.204

Omnibus:	60.022	Durbin-Watson:	2.081
Prob(Omnibus):	0.000	Jarque-Bera (JB):	148.679
Skew:	-1.323	Prob(JB):	5.19e-33
Kurtosis:	6.292	Cond. No.	425.

This model performs pretty well. It accounts for $R^2 = 90\%$ of the variance in the data.

```
plt.rcParams['figure.figsize'] = (15, 9)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.scatter(xs=advert['TV'], ys=advert['Radio'], zs=advert['Sales'])

x = np.linspace(advert['TV'].min(), advert['TV'].max(), 100)
y = np.linspace(advert['Radio'].min(), advert['Radio'].max(), 100)
X,Y = np.meshgrid(x,y)
par = dict(ad_TR_ols.params)
Z = par["Intercept"] + par["TV"]*X + par["Radio"]*Y
surf = ax.plot_surface(X, Y, Z,cmap=cm.Greys, alpha=0.2)

ax.view_init(25,-71)

ax.set_xlabel('TV budget')
ax.set_ylabel('Radio budget')
ax.set_zlabel('Sales')

plt.show()
```


0
50
100
150
200
250
300
TV budget
0
10
20
30
40
50
Radio budget
5
10
15
20
25
Sales

Nonlinear relationships

We can consider the interaction between TV and Radio advertising in the model, by taking

$$\text{Sales} = \beta_0 + \beta_1 * \text{TV budget} + \beta_2 * \text{Radio budget} + \beta_3 \text{TV budget} * \text{Radio budget}.$$

The rational behind the last term is that perhaps spending x on television advertising and y on radio advertising leads to more sales than simply $x + y$. In marketing this is known as the *synergy effect* and in statistics it is known as the *interaction effect*.

Note: even though the relationship between the independent and dependent variables is nonlinear, the model is still linear.

```python
ad_NL = sm.ols(formula="Sales ~ TV + Radio + TV*Radio", data=advert).fit()
ad_NL.summary()
```

OLS Regression Results

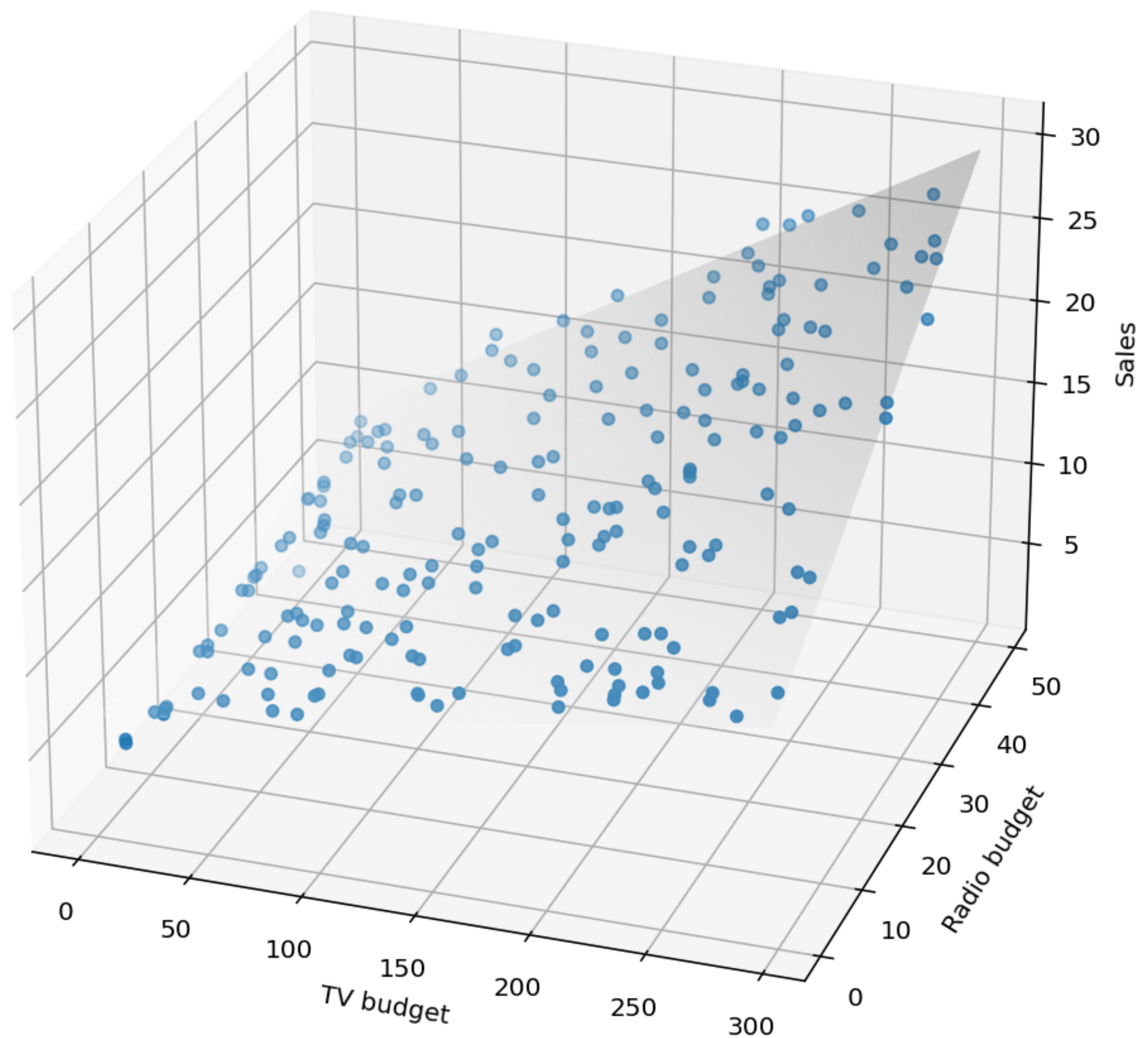
Dep. Variable:	Sales	R-squared:	0.968
Model:	OLS	Adj. R-squared:	0.967
Method:	Least Squares	F-statistic:	1963.
Date:	Wed, 06 Apr 2022	Prob (F-statistic):	6.68e-146
Time:	16:52:02	Log-Likelihood:	-270.14
No. Observations:	200	AIC:	548.3
Df Residuals:	196	BIC:	561.5
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	6.7502	0.248	27.233	0.000	6.261	7.239
TV	0.0191	0.002	12.699	0.000	0.016	0.022
Radio	0.0289	0.009	3.241	0.001	0.011	0.046
TV:Radio	0.0011	5.24e-05	20.727	0.000	0.001	0.001

Omnibus:	128.132	Durbin-Watson:	2.224
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1183.719
Skew:	-2.323	Prob(JB):	9.09e-258
Kurtosis:	13.975	Cond. No.	1.80e+04

This model is really excellent. All of the p-values are small and $R^2 = 97\%$ of the variability in the data is accounted for by the model.

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.scatter(xs=advert['TV'], ys=advert['Radio'], zs=advert['Sales'])

x = np.linspace(advert['TV'].min(), advert['TV'].max(), 100)
y = np.linspace(advert['Radio'].min(), advert['Radio'].max(), 100)
X,Y = np.meshgrid(x,y)
par = dict(ad_NL.params)
Z = par["Intercept"] + par["TV"]*X + par["Radio"]*Y + par["TV:Radio"]*X*Y
surf = ax.plot_surface(X, Y, Z,cmap=cm.Greys, alpha=0.2)

ax.view_init(25,-71)

ax.set_xlabel('TV budget')
ax.set_ylabel('Radio budget')
ax.set_zlabel('Sales')

plt.show()
```

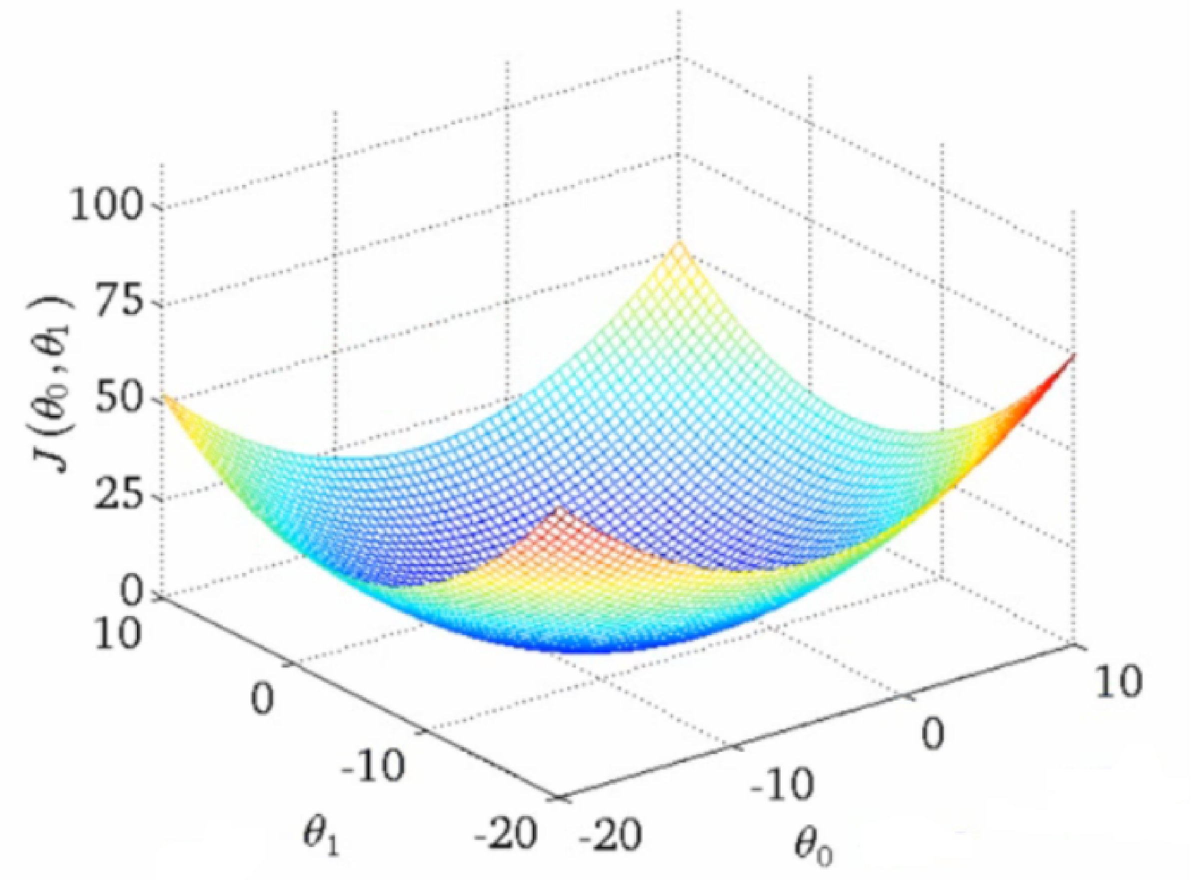

30
25
20
15
10
5
Sales
50
40
30
20
10
0
Radio budget
0
50
100
150
200
250
300
TV budget

A word of caution on overfitting

It is tempting to include a lot of terms in the regression, but this is problematic. A useful model will *generalize* beyond the data given to it.

Gradient Descent

- LinearRegression tries to minimize RSS using Gradient Descent.
- The objective of Gradient Descent is the obtain best weights such that RSS is minimal.



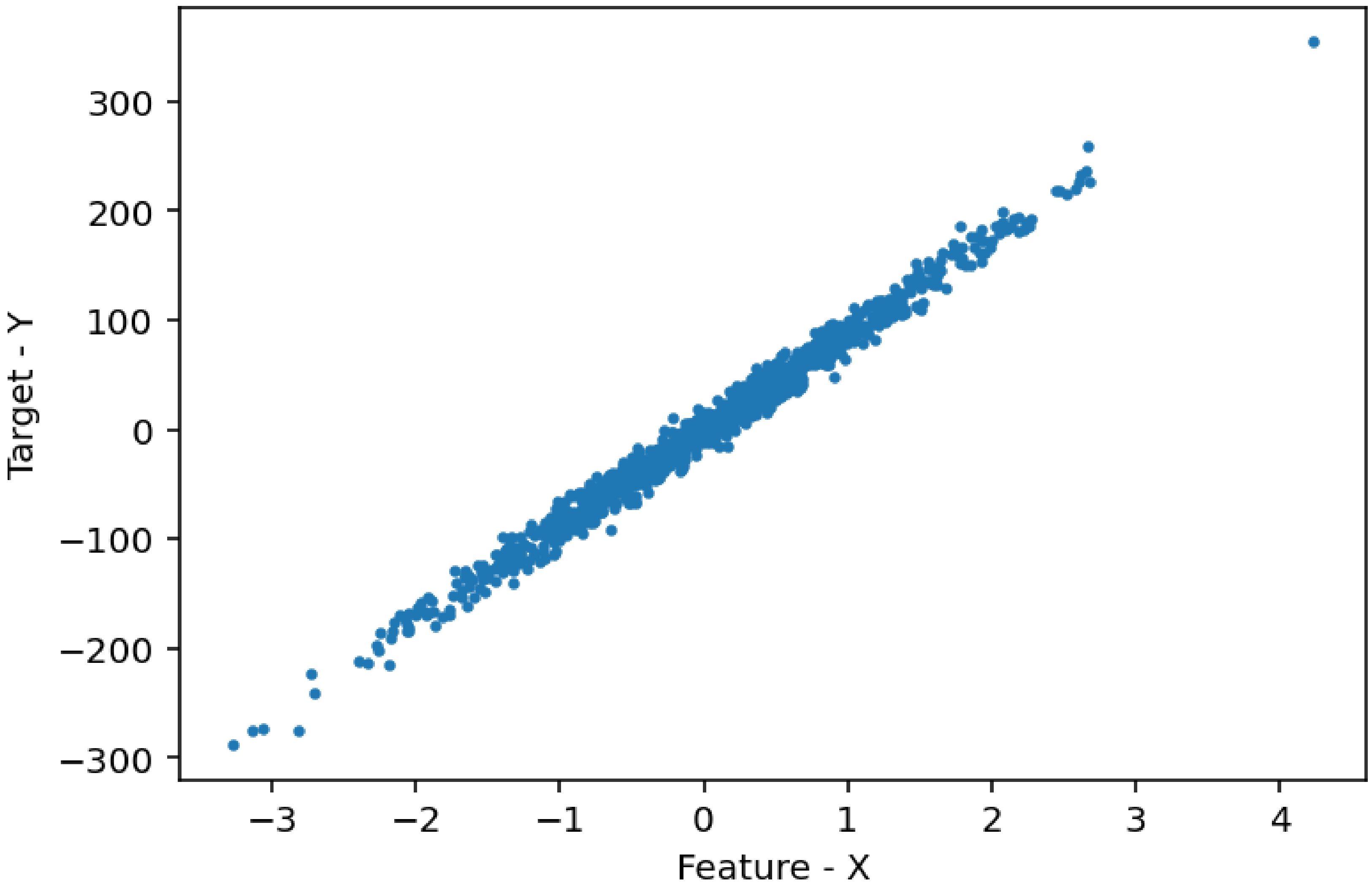
Understanding Math behind gradient descent

- Prediction, $y_p = Ax + B$
- Actual, y
- Simplified Loss for caclulation, Loss = $1/2 * \sum(y_p - y)^2$
- Algorithm
  - Randomly initialize weights A & B
  - Calculate gradient .i.e change in Loss when A & B are changed.
  - Change weights by gradients calculated & reduce the loss
  - Repeat the whole process till weights don't significantly reduce any further

Genrating Regression Dataset

- n_features - number of features to be considered
- noise - deviation from straight line
- n_samples - number of samples

```
from sklearn.datasets import make_regression
X,Y = make_regression(n_features=1, noise=10, n_samples=1000)

plt.xlabel('Feature - X')
plt.ylabel('Target - Y')
plt.scatter(X,Y,s=5)
```


300
200
100
0
−100
−200
−300
Target - Y
−3
−2
−1
0
1
2
3
4
Feature - X

Linear Regression Model

```
lr = LinearRegression()
```

Common Hyperparameters

- fit_interceprt - Whether to calculate intercept for the model, not required if data is centered
- normalize - X will be normalized by subtracting mean & dividing by standard deviation
- By stanrdadizing data before subjecting to model, coef's tells the importance of features

Common Attributes

- coef - weights for each independent variables
- intercept - bias of independent term of linear models

Linear Regression Model

Common Functions

- fit - trains the model. Takes X & Y
- predict - Once model is trained, for given X using predict function Y can be predicted

Multiple Target

- Y can be of more than 1 dimension
- Advantages of multiple target are
  - computationally fast
  - model is optimized for multiple targets
  - model do not use relationship between targets
  - model is more interpretable

Training model

- X should be in rows of data format, X.ndim == 2
- Y should be 1D for simgle target & 2D for more than one target
- fit function for training the model

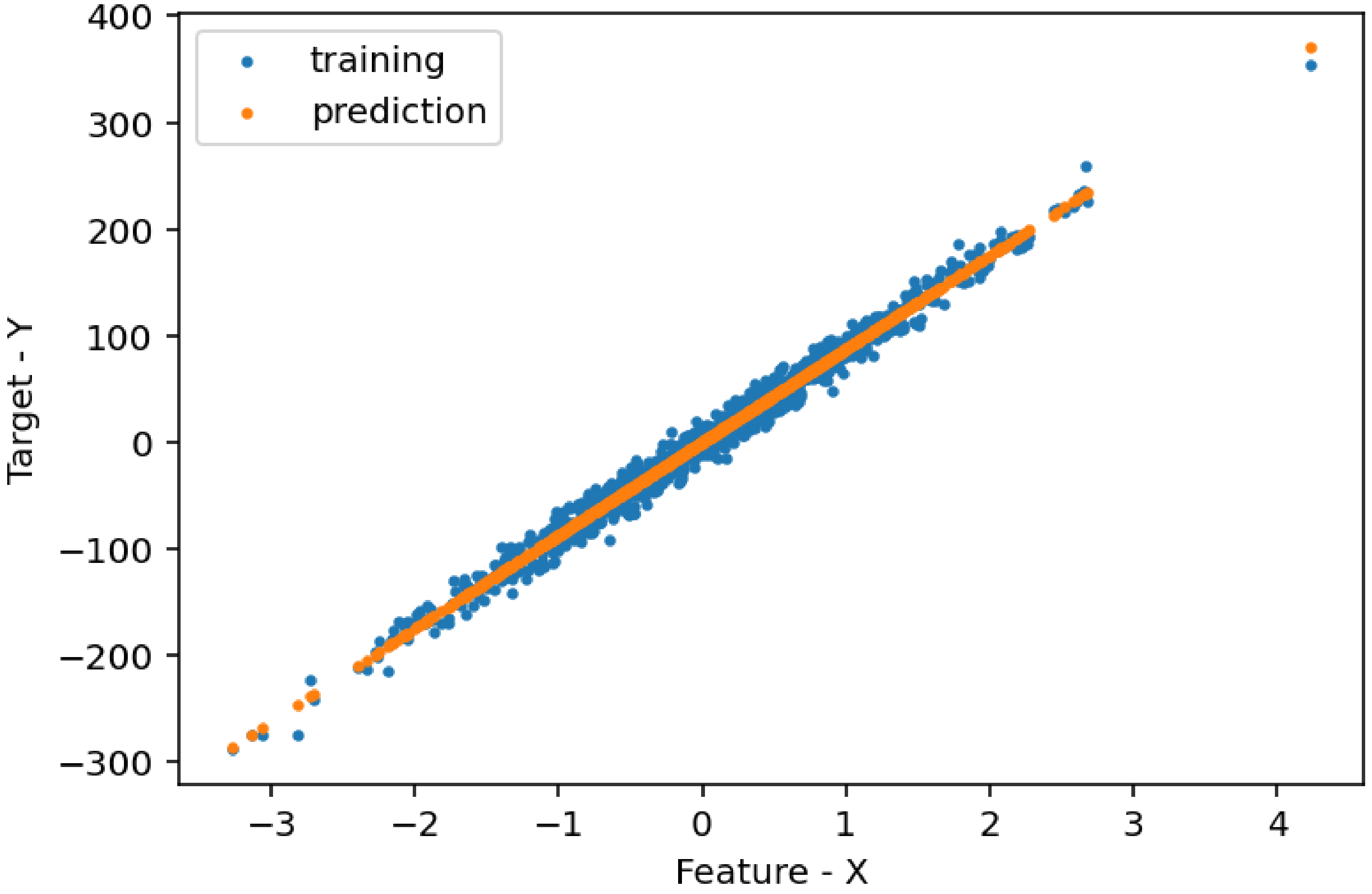
```
lr.fit(X,Y)

print(lr.coef_)
print(lr.intercept_)
```

array([87.40006566])

0.1401629414816563

Predicting using trained model

```
pred = lr.predict(X)

plt.scatter(X,Y,s=5, label='training')
plt.scatter(X,pred,s=5, label='prediction')
plt.xlabel('Feature - X')
plt.ylabel('Target - Y')
plt.legend()
plt.show()
```


training
prediction
400
300
200
100
0
−100
−200
−300
Target - Y
−3
−2
−1
0
1
2
3
4
Feature - X

Limitation of Ordinary Least Square Technique

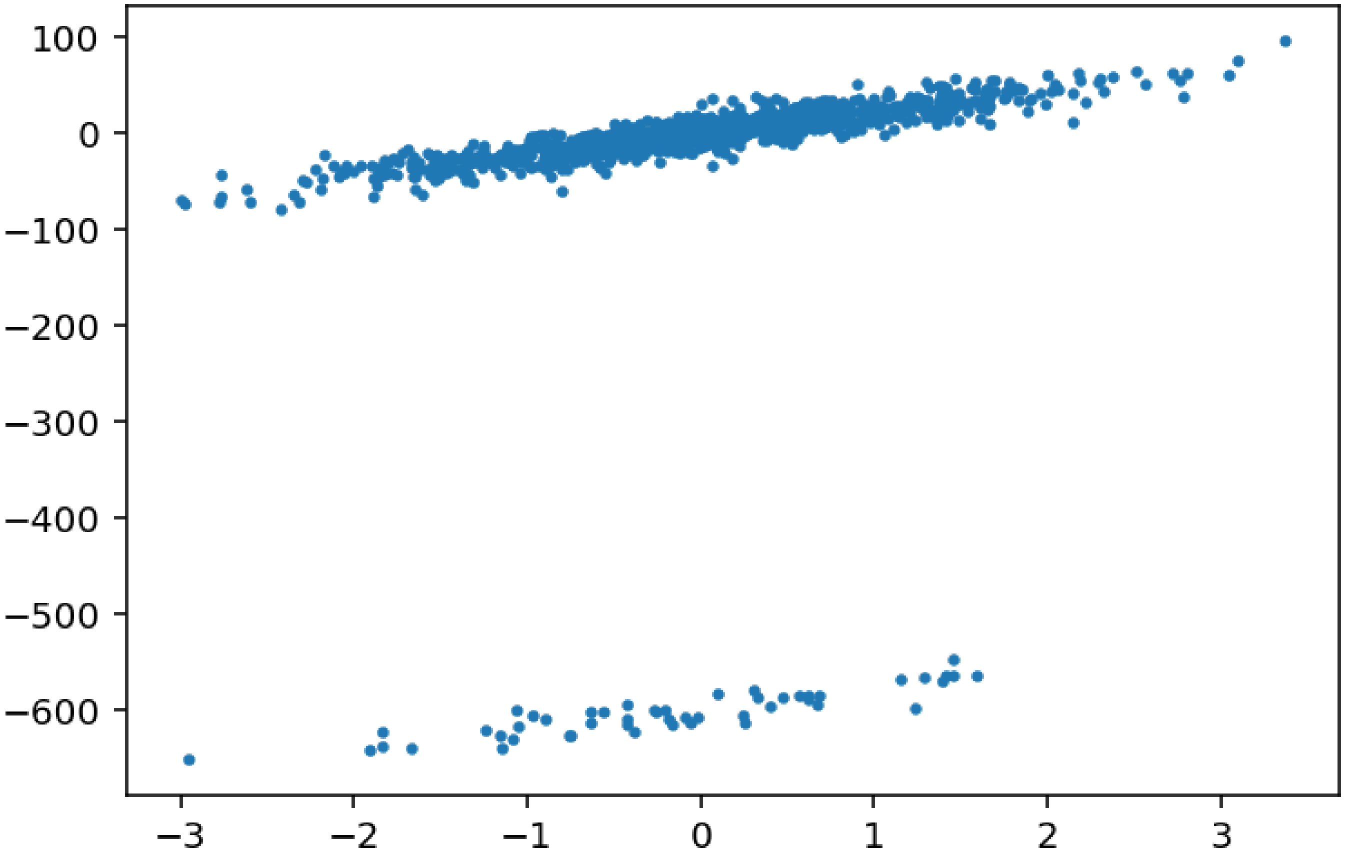
- Impacted by Outliers
- Non-linearities
- Too many independent variables
- Multicollinearity
- Heteroskedasticity
- Noise in the Independent Variables

Ridge Regression

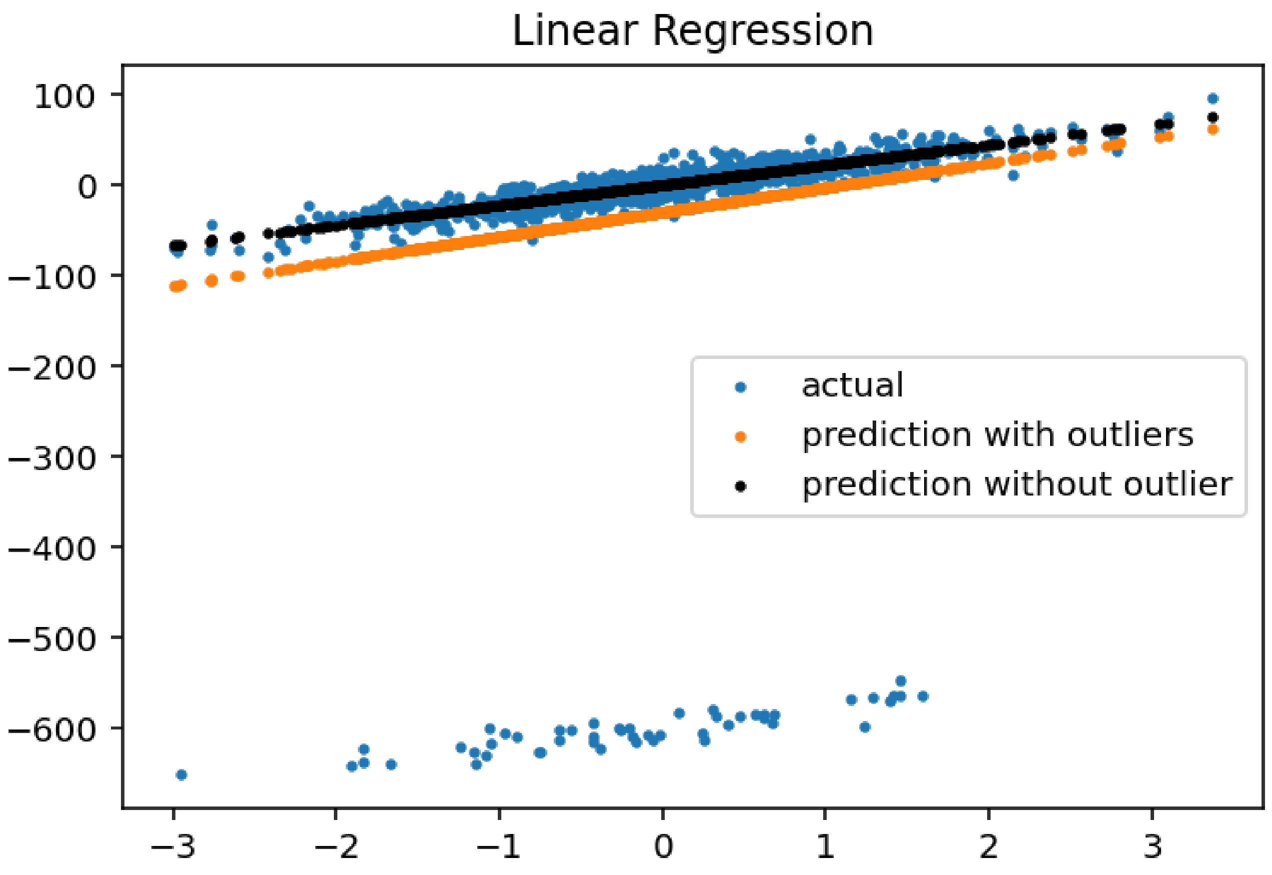
- Ridge Regression imposes penalty on size of coef.
- Less impacted by outliers.

Adding outliers to data

```
outliers = Y[950:] - 600
Y_Out = np.append(Y[:950],outliers)
plt.scatter(X,Y_Out,s=5)
```



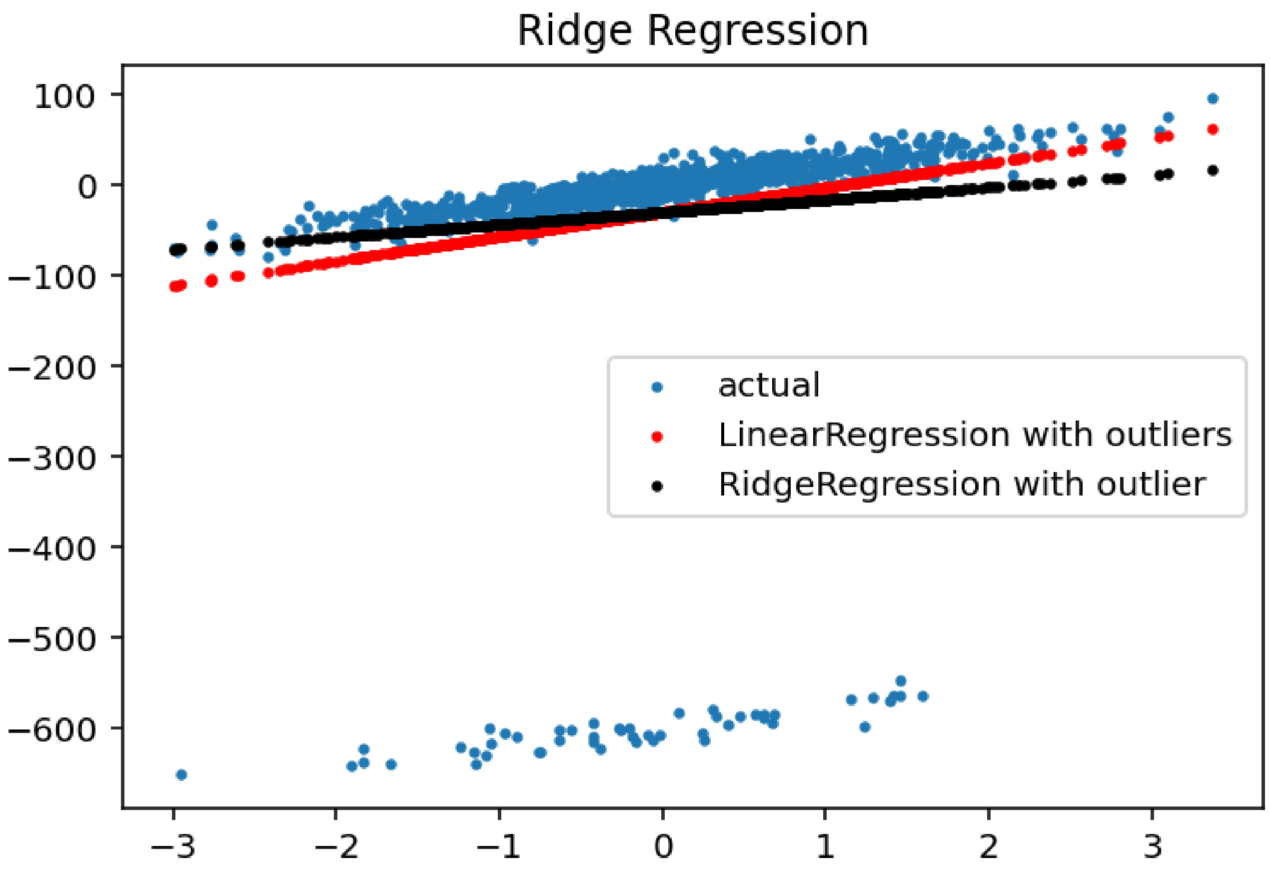
Linear Regression with outliers



Ridge Regression

```
lr = LinearRegression()
lr.fit(X,Y_Out)
pred_Out = lr.predict(X)

ridge = Ridge(alpha=1000)
ridge.fit(X,Y_Out)
pred_ridge = ridge.predict(X)

plt.scatter(X,Y_Out,s=5,label='actual')
plt.scatter(X,pred_Out,s=5, c='r' ,label='LinearRegression with outliers')
plt.scatter(X,pred_ridge,s=5,c='k', label='RidgeRegression with outlier')
plt.legend()
plt.title('Linear Regression')
```

Ridge Regression

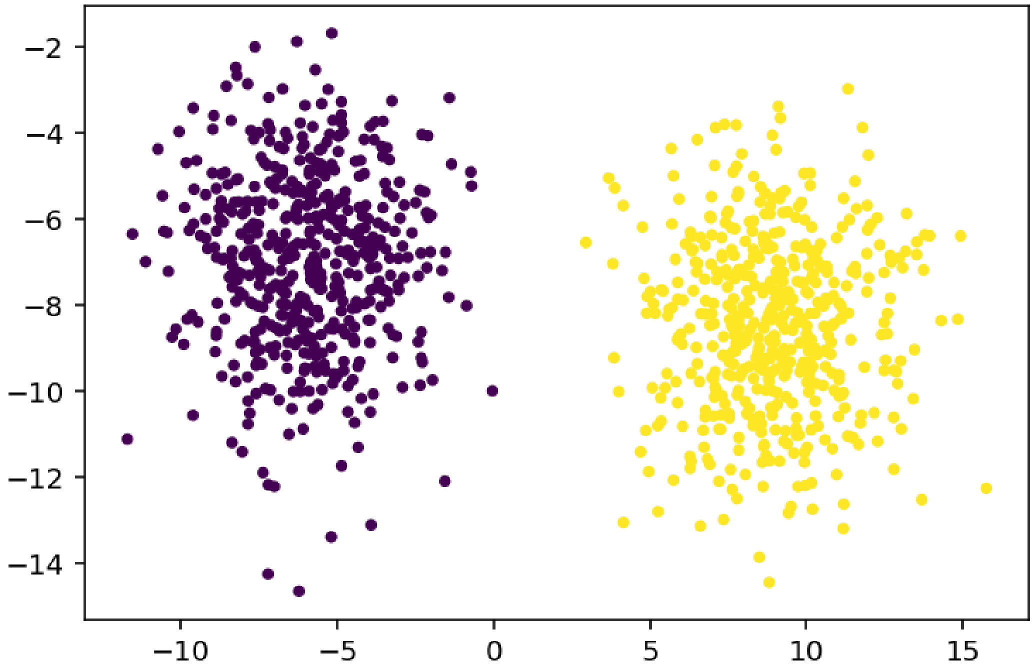


Logistic Regression

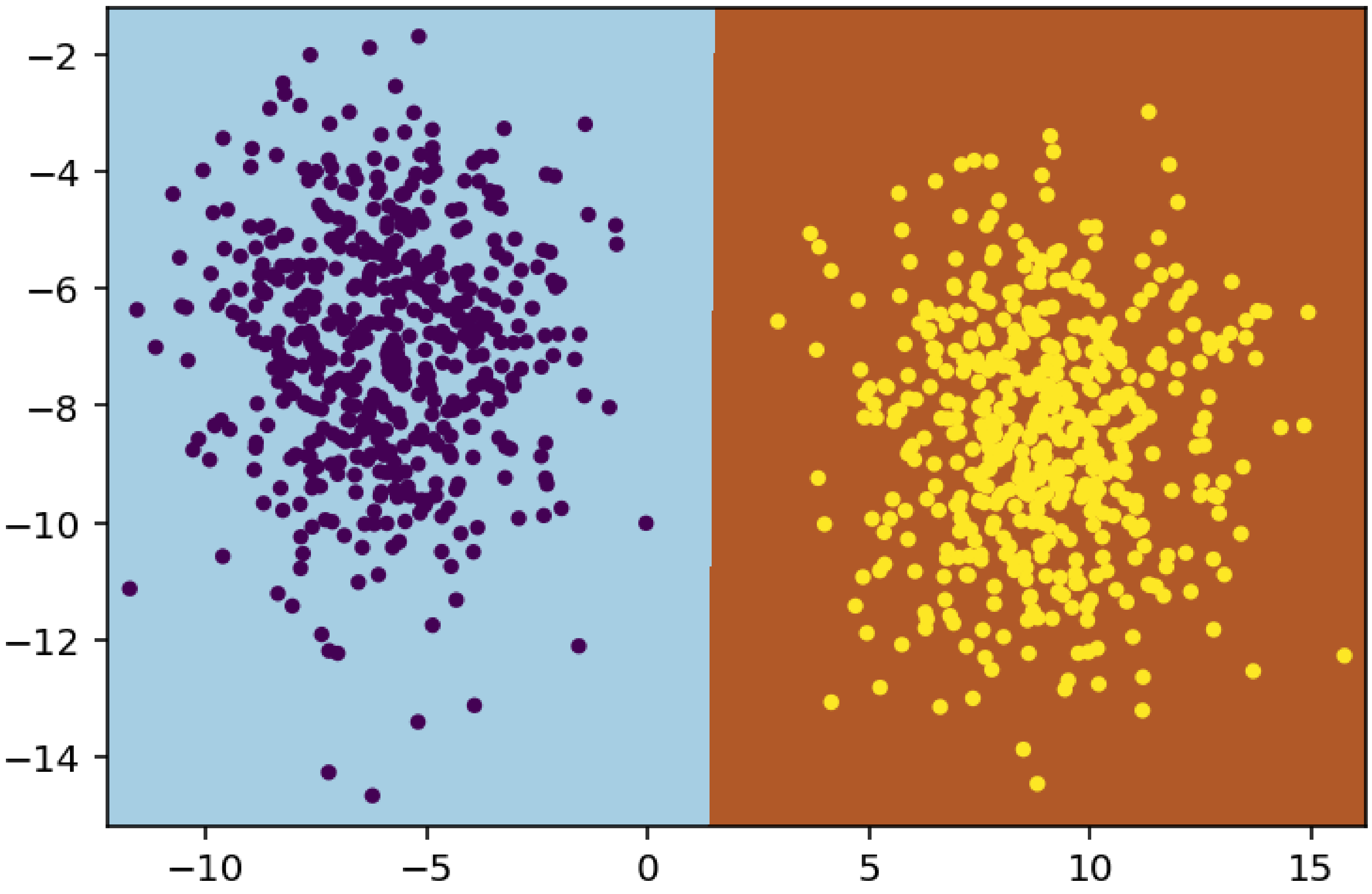
- Linear Model of classification, assumes linear relationship between feature & target
- $y = e^{(b0 + b1x)} / (1 + e^{(b0 + b1x)})$
- Returns class probabilities
- Hyperparameter : C - regularization coef
- Fundamentally suited for bi-class classification

Logistic Regression

```
from sklearn.datasets import make_blobs
X,y = make_blobs(n_features=2, n_samples=1000, cluster_std=2,centers=2)
plt.scatter(X[:,0],X[:,1],c=y,s=10)
```



Logistic Regression



Online Learning Models

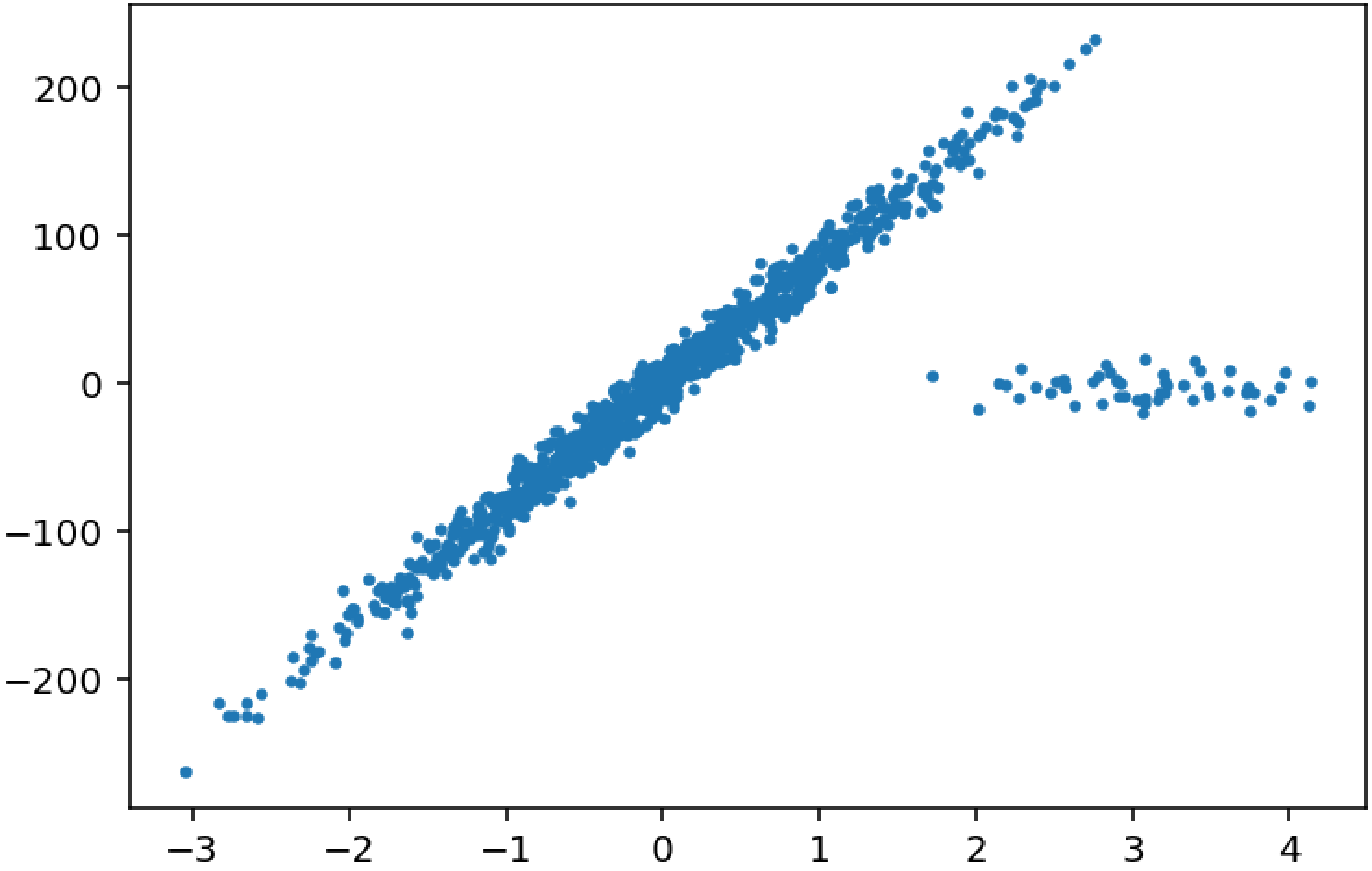
- Stochastic Gradient Descent & Passive Aggrasive Algorithms
- Simple & Efficient to fit linear models
- Useful where number of samples is very large (Scale of 10^5)
- Supports partial_fit for out-of-core learning
- Both the algorithms support regression & classification

Robust Regression

- Robust regression is interested in fitting a regression model in the presence of corrupt data: either outliers, or error in the model.
- Three techniques supported by scikit - RANSAC, Theil Sen and HuberRegressor

```
n_samples = 1000
n_outliers = 50
X, y, coef = make_regression(n_samples=n_samples, n_features=1,
                                      n_informative=1, noise=10,
                                      coef=True, random_state=0)
# Add outlier data
np.random.seed(0)
X[:n_outliers] = 3 + 0.5 * np.random.normal(size=(n_outliers, 1))
y[:n_outliers] = -3 + 10 * np.random.normal(size=n_outliers)

plt.scatter(X,y,s=5)
```

Robust Regression



Robust Regression

```
from sklearn.linear_model import LinearRegression,RANSACRegressor

lr = LinearRegression()
lr.fit(X, y)
ransac = RANSACRegressor()
ransac.fit(X, y)

lr_pred = lr.predict(X)
ransac_pred = ransac.predict(X)

plt.scatter(X,y,s=5, label='data')
plt.scatter(X,ransac_pred,s=5,label='ransac')
plt.scatter(X,lr_pred,s=5, label='linear-regression')
plt.legend()
```

Robust Regression