Linear Regression

Elsa Ferreira Gomes

2024/2025

(https://hastie.su.domains/ISLP/ISLP_website.pdf - Chp3)

Hint: you have an exercise on Moodle for this class, but first, let's review some topics from the last lecture

Simple Linear Regression

How to find an approximated function

Regression is a simple approach to supervised learning, where the dependence of Y on $x_1, x_2, \dots x_N$ is 'linear'.

Consider the simple linear regression - using a single predictor X

$$Y = \beta_0 + \beta_1 X + \epsilon$$

Where β_0 (intercept) and β_1 (slope) are two unknown constants (coefficients or parameters), and ϵ the error.

Estimating β_0 and β_1,

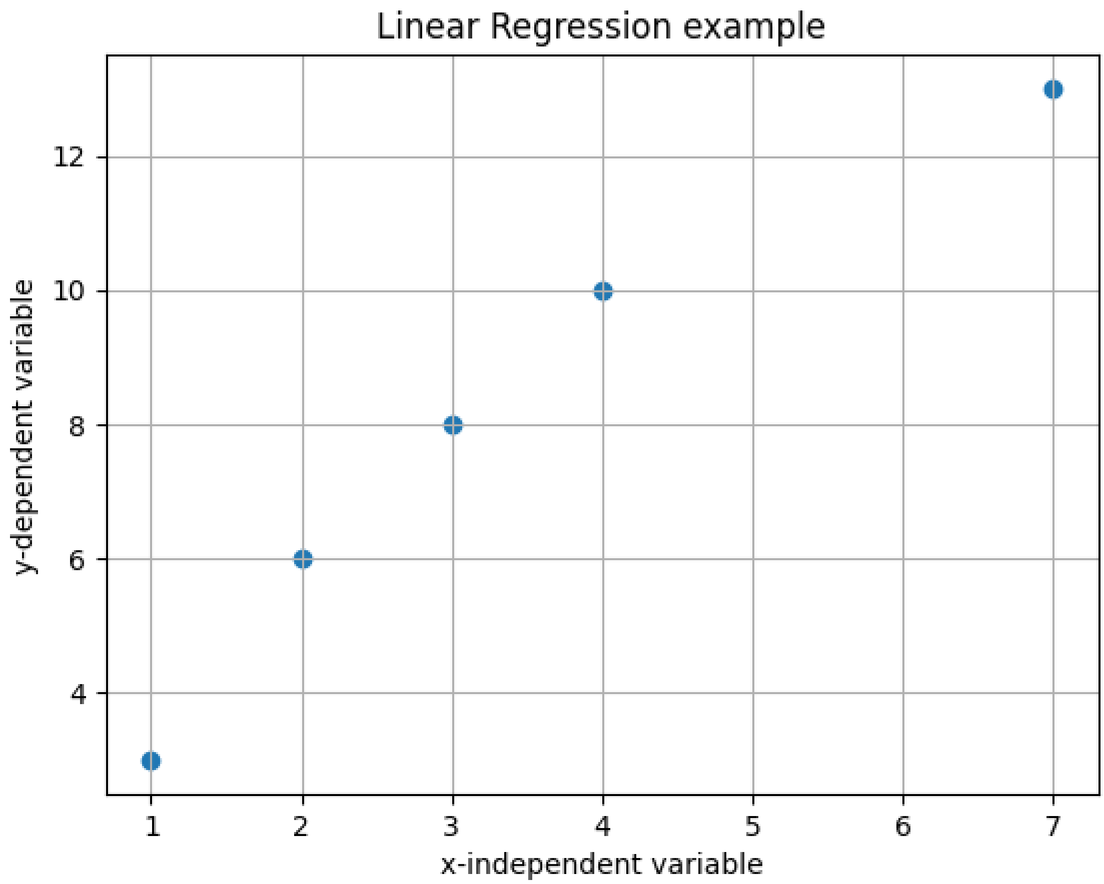
$$\hat{y} = \hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X$$

A simple example using Python

Step1: Visualize data before building the Linear Regression model. Let´s plot the data points to visualize the relashioship between x and y.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
X=[1,2,3,4,7]
Y=[3,6,8,10,13]
plt.scatter(X,Y)
plt.xlabel('x-independent variable')
plt.ylabel('y-dependent variable')
plt.title('Linear Regression example')
plt.grid(True)
plt.show()
```



Step2: Build the Linear Regression model, using scikit-learn library

In [3]:
```
#from sklearn.linear_model import LinearRegression
import sklearn.linear_model as skl_lm
model = skl_lm.LinearRegression()
```

In [4]:
```
X2=[[x] for x in X]
Y=[y for y in Y]
model.fit(X2,Y)
```

Out[4]:
```
▾ LinearRegression
LinearRegression()
```

In [5]:
```
b1=model.coef_[0]
b0=model.intercept_
print("Slope (b1):", b1)
print("Y-intercept (b0):", b0)
```

```
Slope (b1): 1.60377358490566
Y-intercept (b0): 2.5471698113207557
```

Step 3: Make predictions for new values of X

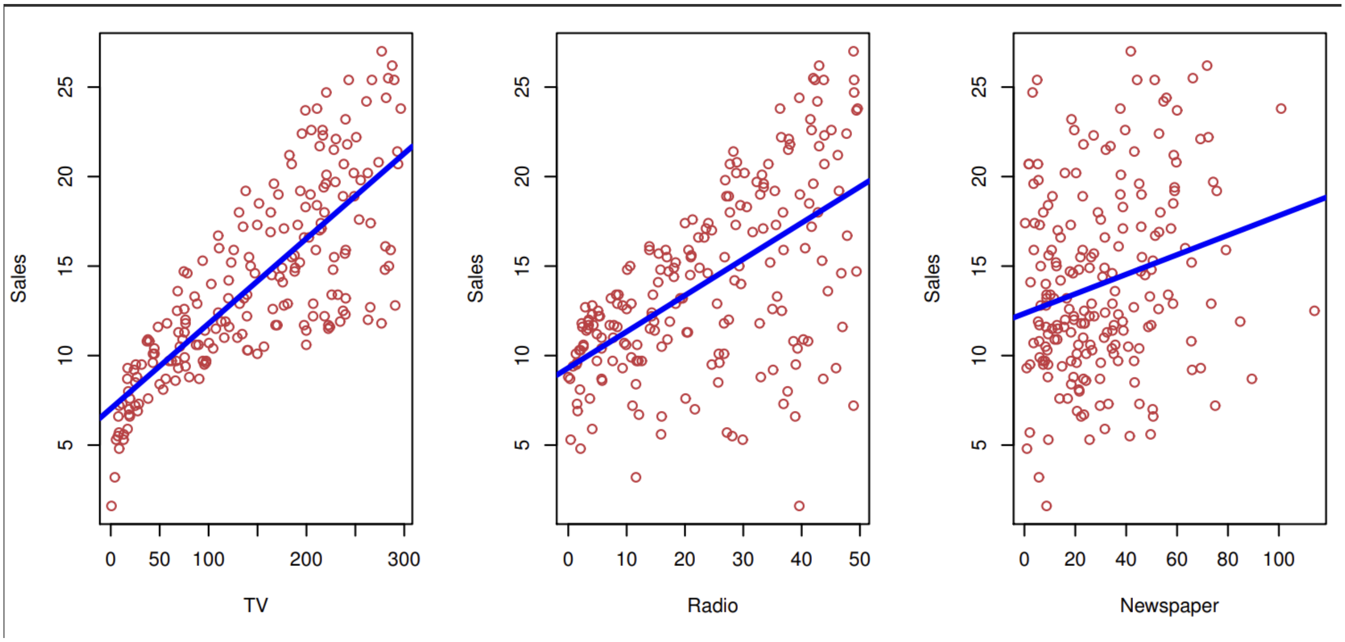
In [6]:
```
new_X=[[6],[7]]
predictions=model.predict(new_X)
print("Prediction for X=6: {0:.4f}".format(predictions[0]))
print("Prediction for X=7:", predictions[1])
```

```
Prediction for X=6: 12.1698
Prediction for X=7: 13.773584905660375
```

Linear Regression

Example: Advertising data

- The advertising dataset captures the sales revenue generated with respect to advertisement costs across multiple channels like radio, tv and newspapers.
- It is required to understand the impact of ad budgets on the overall sales.



Questions:

- Is there a relationship between advertising budget and sales?
- Is the relationship linear?
- Which media contribute to sales?

Linear Regression

Example: Advertising data - some questions we might ask:

- Is there a relationship between advertising budget and sales?
- Which media contribute to sales?
- How accurately can we predict future sales?
- Is the relationship linear?

Estimating the Coefcients (or parameters) by least squares

Consider the simple linear regression - single predictor X

$$Y = \beta_0 + \beta_1 X + \epsilon$$

We intend to estimate β_0 and β_1,

$$\hat{y} = \hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X$$

where $\hat{y}$ is the prediction of Y on the basis of X = x.

(The hat symbol denotes an estimated value)

- $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$
- **How to measure** how much the predicted $\hat{y}$ are close to the actual y
- The difference $e_i = \hat{y}_i - y_i$ is a **residual** or error
  - Best fit has $e_i = 0$ for all i

RSS - Residual Sum of Squares as loss function $\mathcal{L}$

- the Residual (or Errors) Sum of Squares (RSS) between predictions and the true regression targets

$$RSS = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (f(x_i) - \hat{f}\,(x_i))^2 = \sum_{i=1}^{n} (y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))^2$$

- The **least squares** find the parameters β that **minimize** RSS given the data.

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

$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x}$$

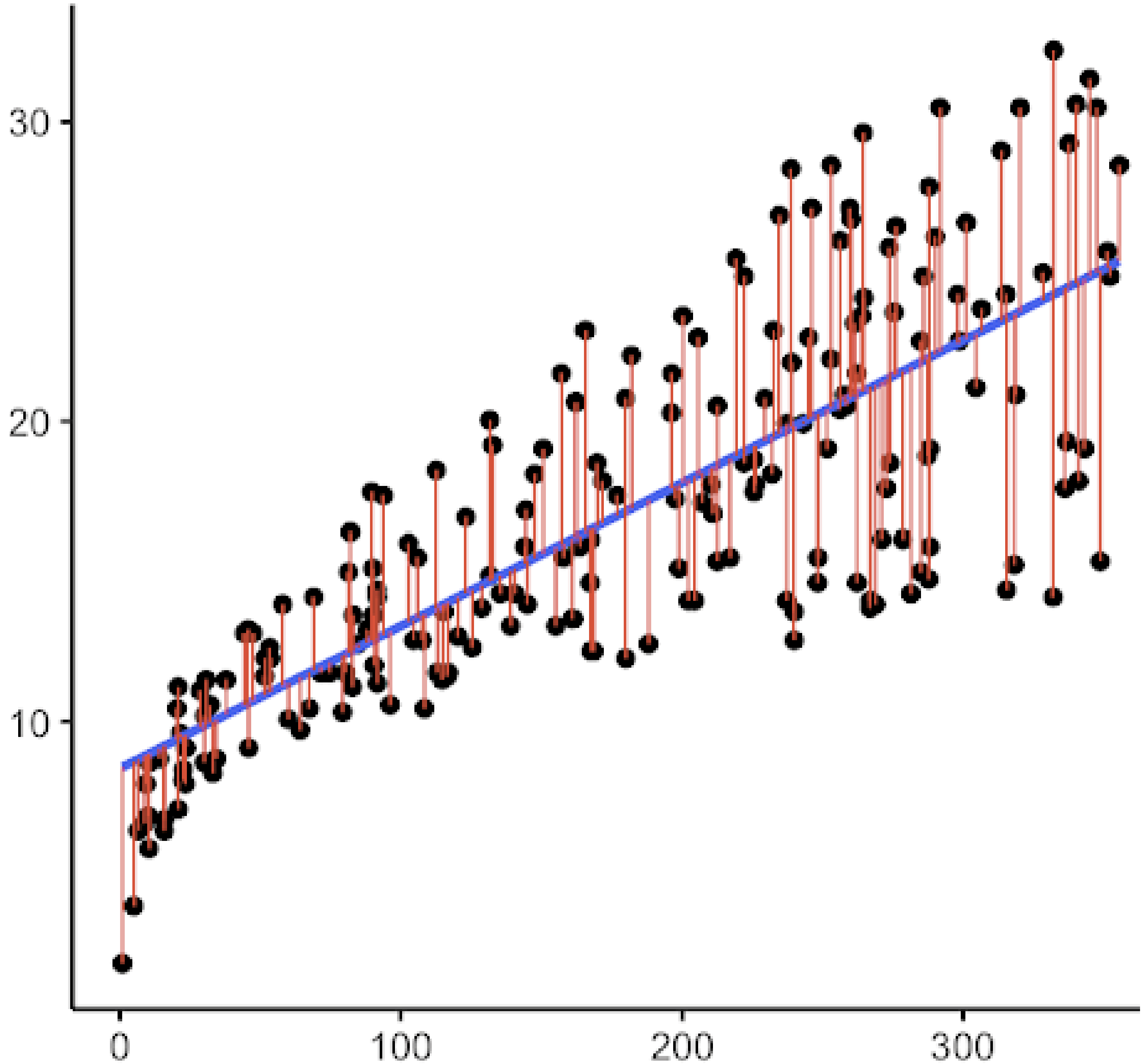
where $\overline{x} = \frac{1}{N} \sum_{i=1}^{n} (x_i)$ and $\overline{y} = \frac{1}{N} \sum_{i=1}^{n} (y_i)$

Example of Residual Sum of Squares - advertising data

The least squares fit for the regression of *sales* onto *TV*.

- **How to measure** how much the predicted $\hat{y}$ are close to the actual y
- The difference $e_i = \hat{y}_i - y_i$ is a **residual** or error.

In the figure above, we can see the least squares regression of sales onto TV for Advertising data, is shown. The fit is obtained by minimizing the some os squared errors. The line segments represents the error.



R^2

How can we measure the intrinsic quality of the model?

- The **sum of the squares** of the **residuals** is a measure of total **error**
  - in the units of Y

$$RSS = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (f(x_i) - \hat{f}(x_i))^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

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

- We **normalize** this error with the total sum of squares $TSS = \sum_{i=1}^{n}(y_i - \overline{y})^2$
- Obtain a **problem independent** measure: R^2

- is a proportion (proportion of variance explained) between 0 and 1 and independent of the scale of Y

Example using Python - Advertising dataset

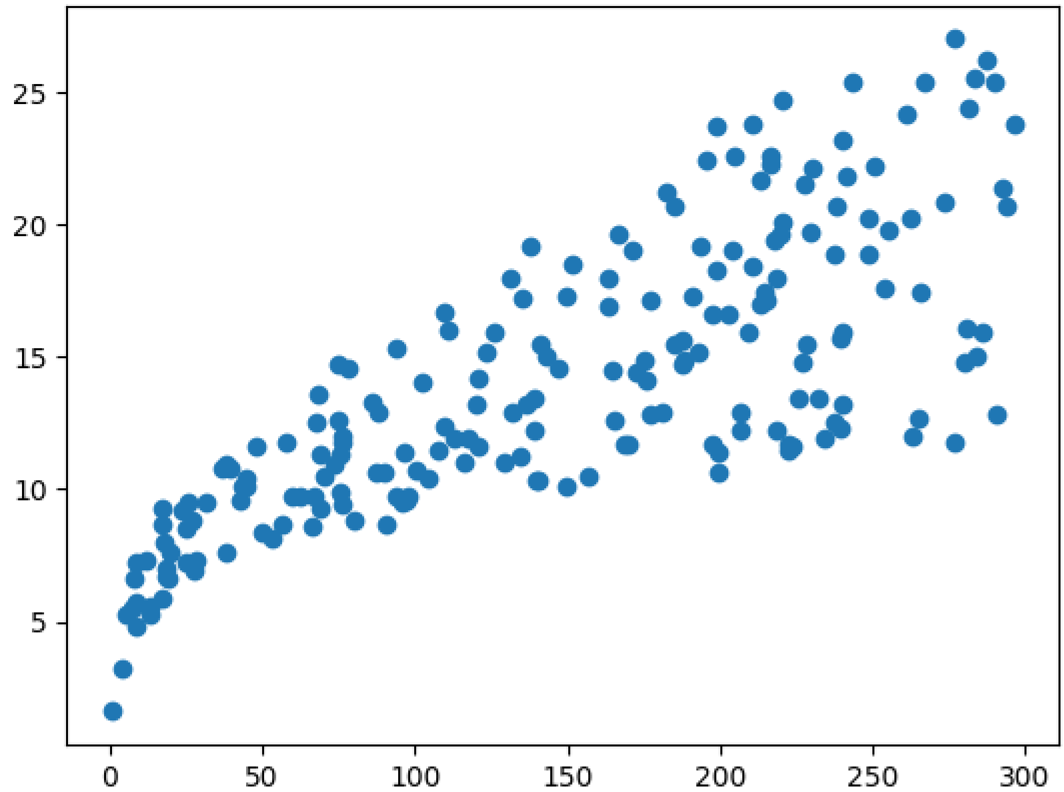
- import libraries
- load data from the csv file

In [7]:
```
#import pandas as pd
#import numpy as np
#import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
import seaborn as sns
from sklearn.preprocessing import scale
import sklearn.linear_model as skl_lm
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.api as sm
import statsmodels.formula.api as smf

###%matplotlib inline plt.style.use('seaborn-white')

advertising = pd.read_csv('Advertising.csv', usecols=[1,2,3,4])
advertising.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   TV         200 non-null    float64
 1   radio      200 non-null    float64
 2   newspaper  200 non-null    float64
 3   sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

In [8]:
```
plt.scatter(x="TV", y="sales", data=advertising)
```

Out[8]:
```
<matplotlib.collections.PathCollection at 0x1b38d90f210>
```



sns.regplot(x="TV", y="sales", data=advertising, order=1, ci=None, scatter_kws={'color':'r', 's':9}) plt.xlim(-10,310) plt.ylim(ymin=0);

In [9]:
```
# Regression coefficients (Ordinary Least Squares)
regr = skl_lm.LinearRegression()

X = scale(advertising.TV, with_mean=True, with_std=False).reshape(-1,1)
y = advertising.sales

regr.fit(X,y)
print(regr.intercept_)
print(regr.coef_)
```

```
14.0225
[0.04753664]
```

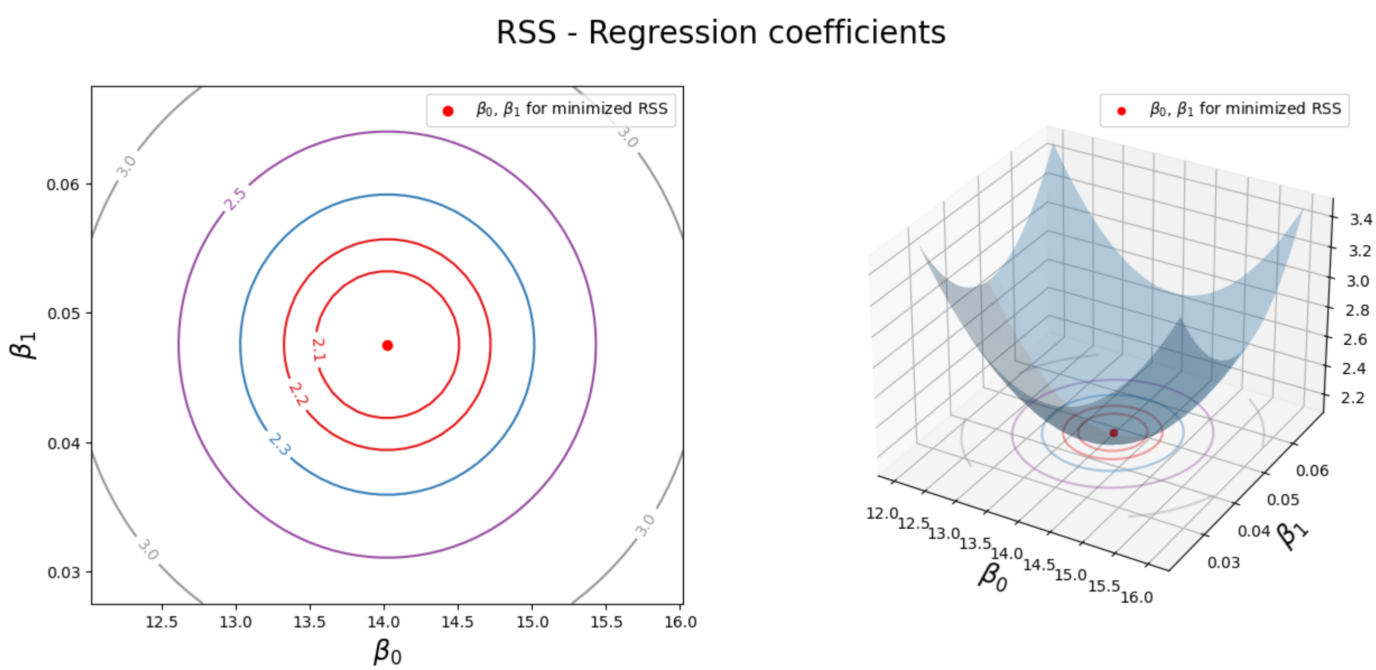
Least squares fit

- Plot the data and the model (based on centered data)
- Consider de model $sales = \beta_0 + \beta_1 TV + \epsilon$

In [10]:
```
# Create grid coordinates for plotting
B0 = np.linspace(regr.intercept_-2, regr.intercept_+2, 50)
B1 = np.linspace(regr.coef_-0.02, regr.coef_+0.02, 50)
xx, yy = np.meshgrid(B0, B1, indexing='xy')
Z = np.zeros((B0.size,B1.size))

# Calculate Z-values (RSS) based on grid of coefficients
for (i,j),v in np.ndenumerate(Z):
    Z[i,j] =((y - (xx[i,j]+X.ravel()*yy[i,j]))**2).sum()/1000

# Minimized RSS
min_RSS = r'$\beta_0$, $\beta_1$ for minimized RSS'
min_rss = np.sum((regr.intercept_+regr.coef_*X - y.values.reshape(-1,1))**2)/1000
min_rss
```

Out[10]:
```
2.1025305831313514
```

In [11]:
```python
fig = plt.figure(figsize=(15,6))
fig.suptitle('RSS - Regression coefficients', fontsize=20)

ax1 = fig.add_subplot(121)
ax2 = fig.add_subplot(122, projection='3d')

# Left plot
CS = ax1.contour(xx, yy, Z, cmap=plt.cm.Set1, levels=[2.15, 2.2, 2.3, 2.5, 3])
ax1.scatter(regr.intercept_, regr.coef_[0], c='r', label=min_RSS)
ax1.clabel(CS, inline=True, fontsize=10, fmt='%1.1f')

# Right plot
ax2.plot_surface(xx, yy, Z, rstride=3, cstride=3, alpha=0.3)
ax2.contour(xx, yy, Z, zdir='z', offset=Z.min(), cmap=plt.cm.Set1,
            alpha=0.4, levels=[2.15, 2.2, 2.3, 2.5, 3])
ax2.scatter3D(regr.intercept_, regr.coef_[0], min_rss, c='r', label=min_RSS)
ax2.set_zlabel('RSS')
ax2.set_zlim(Z.min(),Z.max())
ax2.set_ylim(0.02,0.07)

# settings common to both plots
for ax in fig.axes:
    ax.set_xlabel(r'$\beta_0$', fontsize=17)
    ax.set_ylabel(r'$\beta_1$', fontsize=17)
    ax.set_yticks([0.03,0.04,0.05,0.06])
    ax.legend()
```



In this images, we can see the contour and three-dimensional plots of the RSS obtained using sales as the response and TV as the predictor on the Advertising data. The red hots represent the least squares estimates $\hat{\beta}_0$ and $\hat{\beta}_1$. Where:

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

$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1\overline{x}$$

Confidence interval

H_0: There is no relationship between X and Y

H_A: There is some relationship between X and Y

or,

$$H_0 : \beta_1 = 0$$

$$H_A : \beta_1 <> 0$$

We can compute the $t - test$ to obtain the $p - value$

In [12]:
```
#using statsmodels.formula.api
est = smf.ols('sales ~ TV', advertising).fit()
est.summary().tables[1]
```

Out[12]:

	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

In [13]:
```
# RSS with regression coefficients
((advertising.sales - (est.params[0] + est.params[1]*advertising.TV))**2).sum()/1000
```

Out[13]:
```
2.1025305831313514
```

In [14]:
```
regr = skl_lm.LinearRegression()

X = advertising.TV.values.reshape(-1,1)
y = advertising.sales

regr.fit(X,y)

print("Model intercept:", regr.intercept_)
print("Model slope:     ", regr.coef_)
```

```
Model intercept: 7.032593549127695
Model slope:     [0.04753664]
```

In [15]:
```
Sales_pred = regr.predict(X)
r2_score(y, Sales_pred)
```

Out[15]:
```
0.611875050850071
```

The evaluation of the Model will be discussed in another lecture. We used the data for the test but this procedure is does not apply in general.

Estimating parameters by Least squares

- RSS for the multidimensional case

$$RSS(\beta) = (\mathbf{y} - \mathbf{X}\beta))^T(\mathbf{y} - \mathbf{X}\beta)$$

where $\mathbf{X}$ is the $N \times p$ **data matrix**

If $\mathbf{X}^T\mathbf{X}$ is nonsingular then we have a unique solution for the equation

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Multiple Linear Regression

How to find an approximated function

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \epsilon$$

In the case of our example, we can consider the following model:

$$sales = \beta_0 + \beta_1 TV + \beta_2 radio + \beta_3 newspaper + \epsilon$$

Estimation and Predition

We can use:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \ldots + \hat{\beta}_p x_p$$

RSS - Residual Sum of Squares as loss function

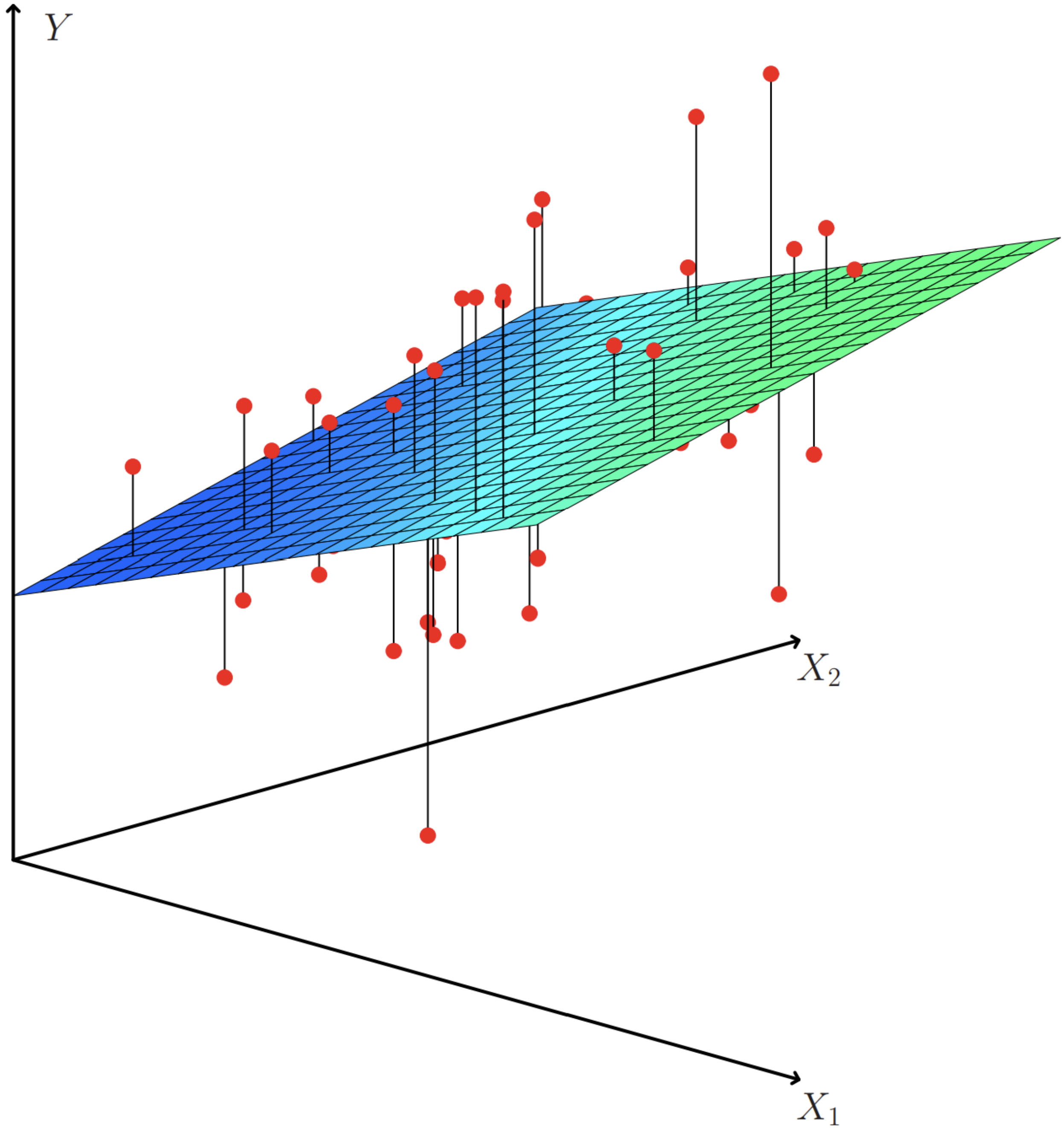
We intend to estimate β_0, β_1,..., β_p that minimize the sum of squared residuals.

$$RSS = \sum_{i=1}^{n} e_i^2 = e_1^2 + e_2^2 + \ldots + e_n^2 \ldots = \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \ldots - \hat{\beta}_p x_{ip})^2$$

In the our example, the model takes the form:

$$sales = \hat{\beta}_0 + \hat{\beta}_1 \times TV + \hat{\beta}_2 \times radio + \hat{\beta}_3 \times newspaper + \epsilon$$

In this figure we can see a three-dimensional setting, with two predictors and one response, the least squares regression line becomes a plane. The plane is chosen to minimize the sum of the squared vertical distances between each observation (in red) and the plane.



In [16]:
```
est = smf.ols('sales ~ TV + radio + newspaper', advertising).fit()
est.summary()
```

Out[16]:

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, 25 Sep 2024	**Prob (F-statistic):**	1.58e-96
Time:	23:10:22	**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.

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

Potential Problems

When we fit a linear regression model to dataset, many problems may occur:

1. Non-linearity of the response-predictor relationships.

1. Correlation of error terms.

1. Non-constant variance of error terms.

1. Outliers.

1. High-leverage points.

1. Collinearity

Evaluation of the correlation amongst the regression predictors

Correlation Matrix

In [17]:
```
advertising.corr()
```

Out[17]:

	TV	radio	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

In [18]:
```
regr = skl_lm.LinearRegression()

X = advertising[['radio', 'TV']].to_numpy()
y = advertising.sales
```

```
regr.fit(X,y)

print(regr.coef_)
print(regr.intercept_)
```

```
[0.18799423 0.04575482]
2.921099912405138
```

In [19]:

```
# What are the min/max values of Radio & TV?
# Use these values to set up the grid for plotting.
advertising[['radio', 'TV']].describe()
```

Out[19]:

	radio	TV
count	200.000000	200.000000
mean	23.264000	147.042500
std	14.846809	85.854236
min	0.000000	0.700000
25%	9.975000	74.375000
50%	22.900000	149.750000
75%	36.525000	218.825000
max	49.600000	296.400000

In [20]:

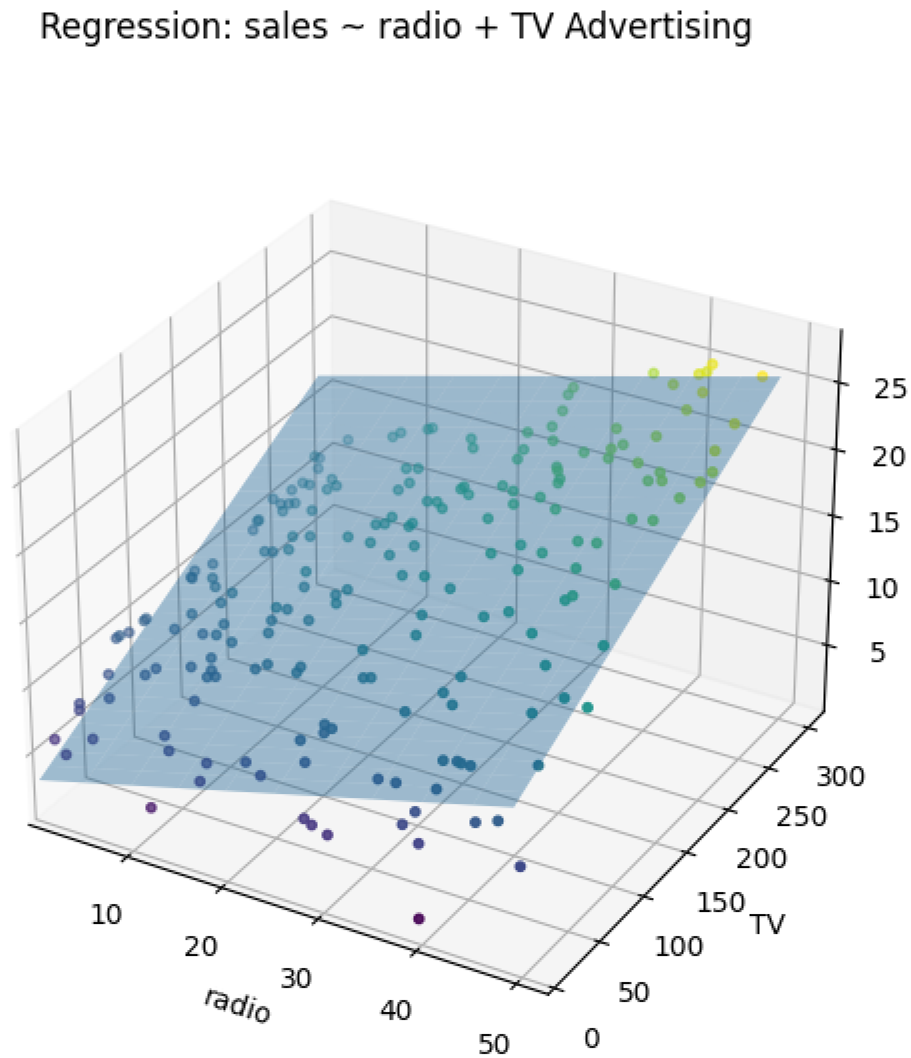
```
# Create a coordinate grid
radio = np.arange(0,50)
TV = np.arange(0,300)

B1, B2 = np.meshgrid(radio, TV, indexing='xy')
Z = np.zeros((TV.size, radio.size))

for (i,j),v in np.ndenumerate(Z):
        Z[i,j] =(regr.intercept_ + B1[i,j]*regr.coef_[0] + B2[i,j]*regr.coef_[1])
```

In [21]:

```
# Create plot
fig = plt.figure(figsize=(10, 6))
fig.suptitle('Regression: sales ~ radio + TV Advertising', fontsize=12)

ax = fig.add_subplot(projection='3d')

ax.plot_surface(B1, B2, Z, rstride=10, cstride=5, alpha=0.4)

ax.scatter3D(advertising.radio, advertising.TV, advertising.sales, c=y, s=10,
           cmap='viridis')
# c='r', s=10)

ax.set_xlabel('radio')
ax.set_xlim(0.50)
ax.set_ylabel('TV')
ax.set_ylim(ymin=0)
ax.set_zlabel('sales')
```

Out[21]:

```
Text(0.5, 0, 'sales')
```



In this three-dimensional image with two predictors (TV and radio) and one response (sales), the least squares regression line becomes a plane. When levels of either TV or radio are low, then the true sales are lower than predicted by the linear model. But when advertising is split between the two media, then the model tends to underestimate sales.

Questions

1. Which predictors X1, X2,...,Xn are useful in predicting the answer.

1. All the predictors are useful.

1. How well the model fit the data.

1. Given a set of predictor values, what response value should we predict, and how accurate is our prediction.

Non-linear relationships

- A very simple way to directly extend the linear model to accommodate non-linear relationships, using polynomial regression.

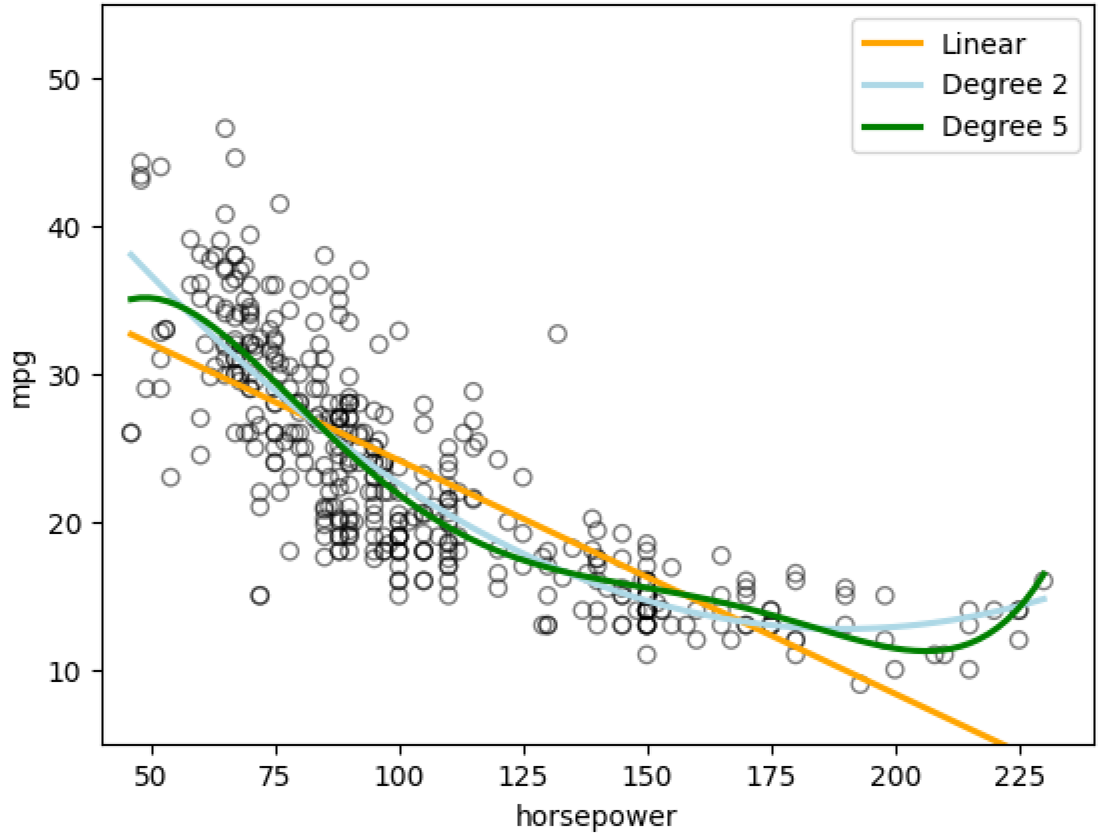
- Consider the example of mpg (gas mileage in miles per gallon) versus horsepower for a number of cars in the Auto data set.

- The orange line represents the linear regression fit.
- There is a pronounced relationship between mpg and horsepower
    - It seems clear that this relationship is in fact non-linear: the data suggest a curved relationship.
    - A simple approach for incorporating non-linear associations in a linear model is to include transformed versions of the predictors.
- For example, the points in to have a quadratic shape, suggesting a model like:

$$mpg = \beta_0 + \beta_1 horsepower + \beta_2 horsepower^2 + \epsilon$$

In [22]:
```
auto = pd.read_csv('Auto.csv', na_values='?').dropna()
auto.info()

# With Seaborn's regplot() you can easily plot higher order polynomials.
plt.scatter(auto.horsepower, auto.mpg, facecolors='None', edgecolors='k', alpha=.5)

sns.regplot(x="horsepower", y="mpg", data=auto, ci=None, label='Linear', scatter=False,
sns.regplot(x="horsepower", y="mpg", data=auto, ci=None, label='Degree 2', order=2, scat
sns.regplot(x="horsepower", y="mpg", data=auto, ci=None, label='Degree 5', order=5, scat

plt.legend()
plt.ylim(5,55)
plt.xlim(40,240);
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 392 entries, 0 to 396
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           392 non-null    float64
 1   cylinders     392 non-null    int64
 2   displacement  392 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        392 non-null    int64
 5   acceleration  392 non-null    float64
 6   year          392 non-null    int64
 7   origin        392 non-null    int64
 8   name          392 non-null    object
dtypes: float64(4), int64(4), object(1)
memory usage: 30.6+ KB
```



In [23]:
```
auto['horsepower2'] = auto.horsepower**2
auto.head(3)
```

Out[23]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name	horsepower2
0	18.0	8	307.0	130.0	3504	12.0	70	1	chevrolet chevelle malibu	16900.0
1	15.0	8	350.0	165.0	3693	11.5	70	1	buick skylark 320	27225.0
2	18.0	8	318.0	150.0	3436	11.0	70	1	plymouth satellite	22500.0

In [24]:
```
est = smf.ols('mpg ~ horsepower + horsepower2', auto).fit()
est.summary().tables[1]
```

Out[24]:

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	56.9001	1.800	31.604	0.000	53.360	60.440
horsepower	-0.4662	0.031	-14.978	0.000	-0.527	-0.405
horsepower2	0.0012	0.000	10.080	0.000	0.001	0.001