1 Homework

1.1 Import

[3]:
```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import style
from sklearn.datasets import fetch_openml, make_classification
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from mpl_toolkits.mplot3d import Axes3D
```

1.2 1.7

[4]:
```
def check_bmi(bmi):
    if bmi < 18.5:
        return 'Underweight'
    elif bmi < 25:
        return 'Normal weight'
    elif bmi < 30:
        return 'Overweight'
```

[5]:
```
names = np.array(['Ann', 'Joe', 'Mark'])
heights = np.array([1.5, 1.78, 1.6])
weights = np.array([65, 46, 59])

bmi = weights / heights ** 2
bmi
```

[5]:
```
array([28.88888889, 14.51836889, 23.046875  ])
```

[6]:
```
df = pd.DataFrame({'Name': names, 'Height': heights, 'Weight': weights, 'BMI':␣
 ↪bmi})
df
```

[6]:

	Name	Height	Weight	BMI
0	Ann	1.50	65	28.888889
1	Joe	1.78	46	14.518369
2	Mark	1.60	59	23.046875

[7]:
```
classify = np.vectorize(check_bmi)
classify(bmi)
```

[7]:
```
array(['Overweight', 'Underweight', 'Normal weight'], dtype='<U13')
```

[8]:
```python
df2 = pd.DataFrame({
  'Name': names,
  'Height': heights,
  'Weight': weights,
  'BMI': bmi,
  'Classify': classify(bmi)})
df2
```

[8]:

	Name	Height	Weight	BMI	Classify
0	Ann	1.50	65	28.888889	Overweight
1	Joe	1.78	46	14.518369	Underweight
2	Mark	1.60	59	23.046875	Normal weight

1.2.1 Data from group

[9]:
```python
data = pd.read_csv('data/no1_7.csv')
data
```

[9]:

	name	height	weight
0	anhnt	1.66	72
1	vphuong	1.78	65
2	vu	1.68	60
3	nam	1.69	65
4	dphuong	1.67	60

[10]:
```python
names = data['name'].values
heights = data['height'].values
weights = data['weight'].values
```

[11]:
```python
bmi = weights / heights ** 2
bmi
```

[11]:
```
array([26.12861083, 20.51508648, 21.2585034 , 22.75830678, 21.51385851])
```

[12]:
```python
df3 = pd.DataFrame({
  'Name': names,
  'Height': heights,
  'Weight': weights,
  'BMI': bmi,
  'Classify': classify(bmi)})
df3
```

[12]:

	Name	Height	Weight	BMI	Classify
0	anhnt	1.66	72	26.128611	Overweight
1	vphuong	1.78	65	20.515086	Normal weight
2	vu	1.68	60	21.258503	Normal weight
3	nam	1.69	65	22.758307	Normal weight
4	dphuong	1.67	60	21.513859	Normal weight

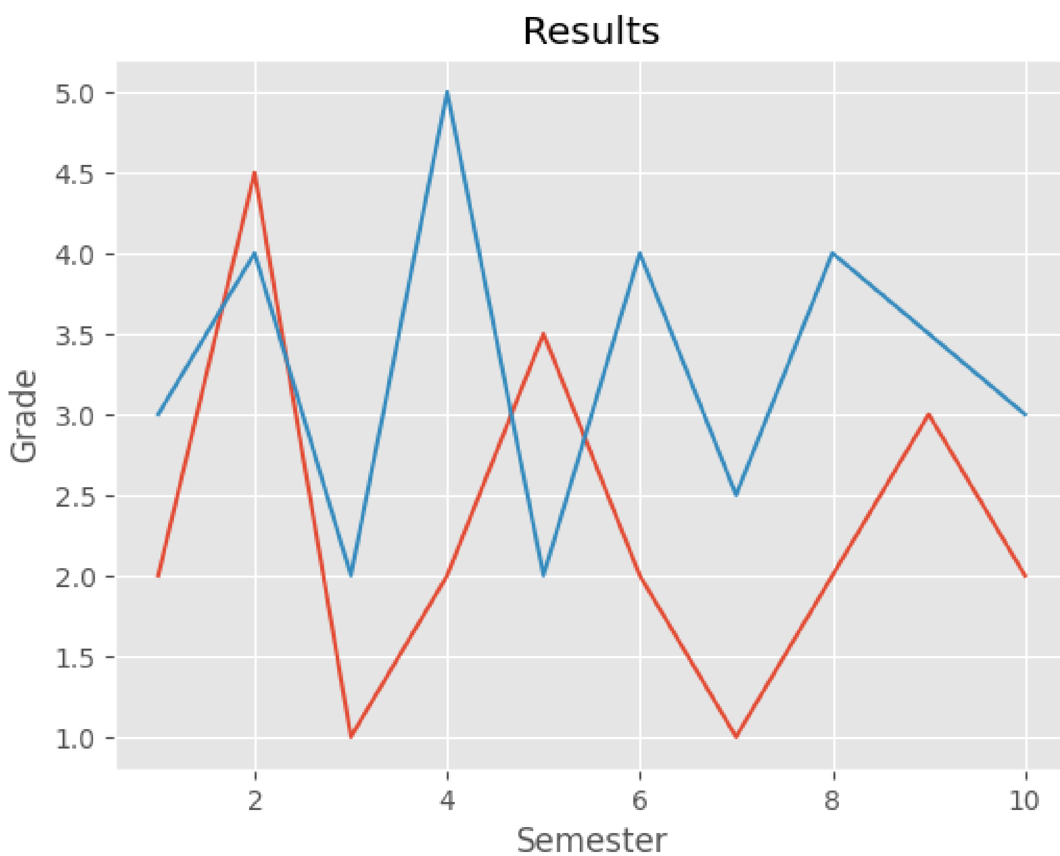
1.3 1.8

Plotting multiple Lines in the same chart

[13]:
```
style.use('ggplot')
```

[14]:
```
plt.plot(
  [1,2,3,4,5,6,7,8,9,10],
  [2,4.5,1,2,3.5,2,1,2,3,2]
)

plt.plot(
  [1,2,3,4,5,6,7,8,9,10],
  [3,4,2,5,2,4,2.5,4,3.5,3]
)

plt.title('Results')
plt.xlabel('Semester')
plt.ylabel('Grade')
```

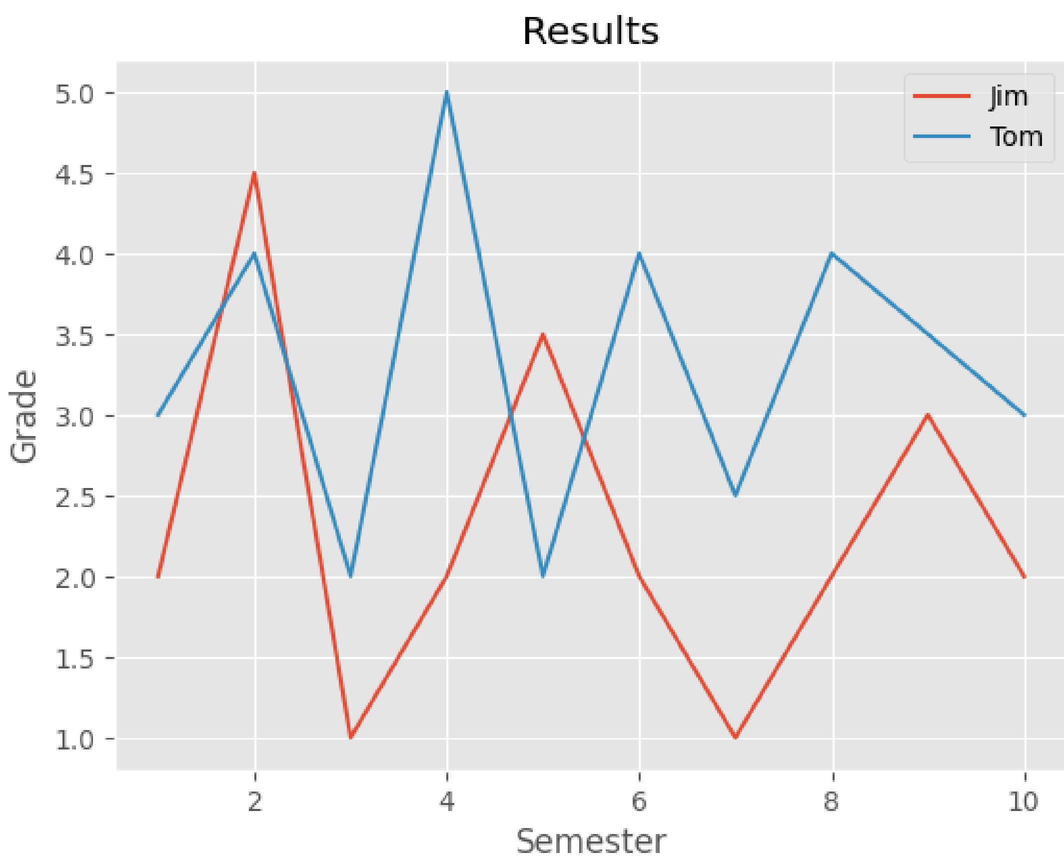
[14]:
```
Text(0, 0.5, 'Grade')
```



Adding a Legend

[15]:
```
plt.plot(
  [1,2,3,4,5,6,7,8,9,10],
  [2,4.5,1,2,3.5,2,1,2,3,2],
  label="Jim"
)
plt.plot(
  [1,2,3,4,5,6,7,8,9,10],
  [3,4,2,5,2,4,2.5,4,3.5,3],
  label="Tom"
)

plt.title('Results')
plt.xlabel('Semester')
plt.ylabel('Grade')
plt.legend()
```

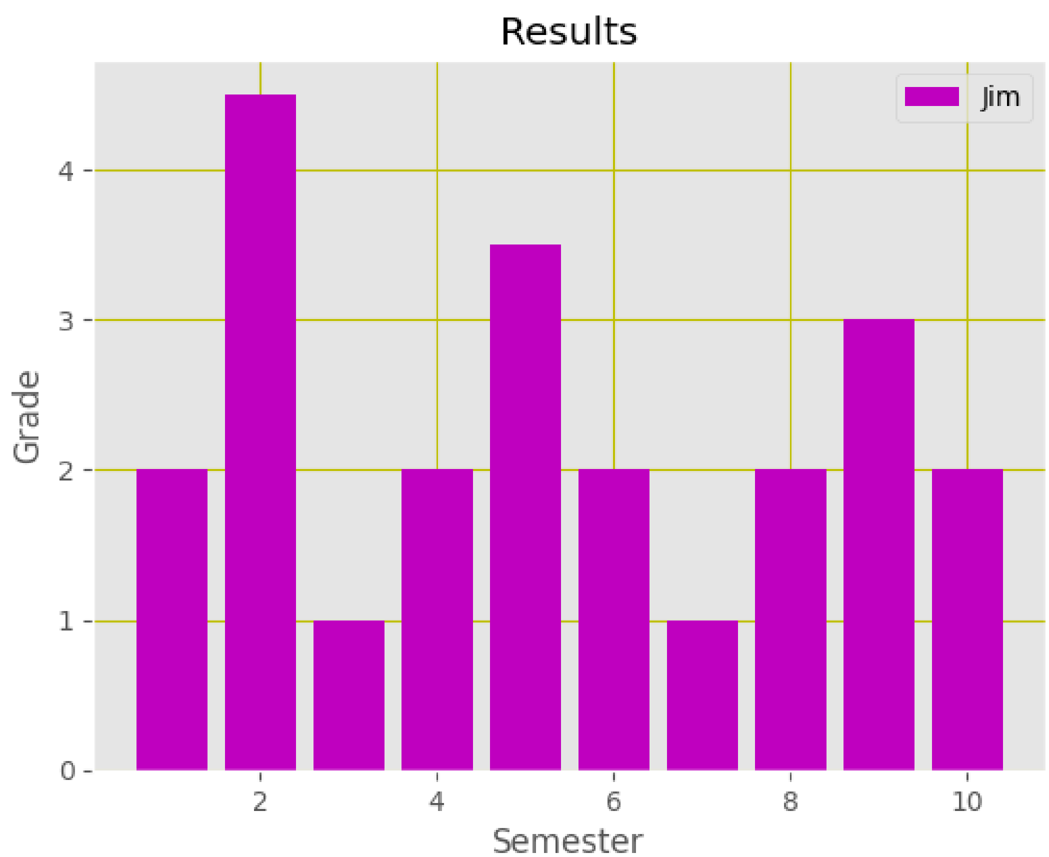
[15]:
```
<matplotlib.legend.Legend at 0x1bfce2274c0>
```



Plotting bar charts

[16]:

```
plt.bar(
    [1,2,3,4,5,6,7,8,9,10],
    [2,4.5,1,2,3.5,2,1,2,3,2],
    label = "Jim",
    color = "m",                    # m for magenta
    align = "center"
)

plt.title("Results")
plt.xlabel("Semester")
plt.ylabel("Grade")

plt.legend()
plt.grid(True, color="y")
```



1.3.1 Data from team

[17]:
```
df = pd.read_csv('./data/no1_8.csv')
df
```

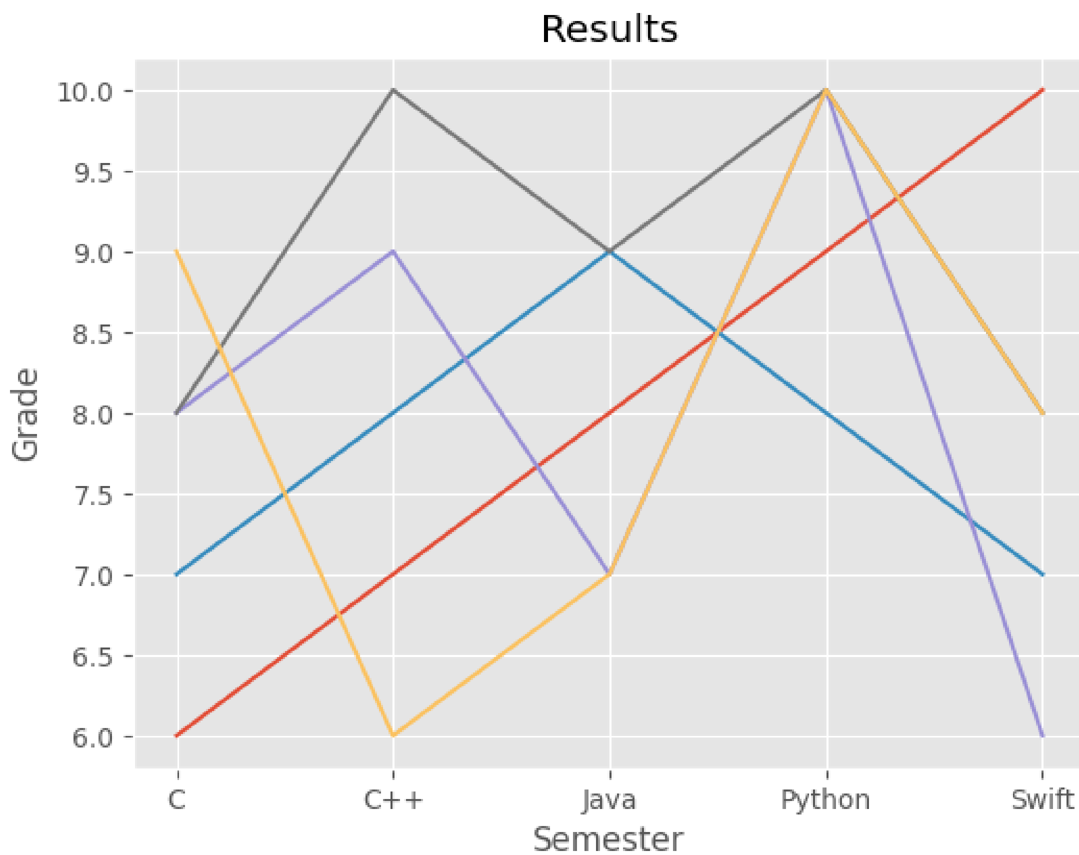
[17]:

	Subject	tienanh	vphuong	vu	nam	dphuong
0	C	6	7	8	8	9
1	C++	7	8	9	10	6
2	Java	8	9	7	9	7
3	Python	9	8	10	10	10
4	Swift	10	7	6	8	8

[18]:
```
subjects = df["Subject"]
tienanh_grade = df["tienanh"]
vphuong_grade = df["vphuong"]
vu_grade = df["vu"]
nam_grade = df["nam"]
dphuong_grade = df["dphuong"]
```

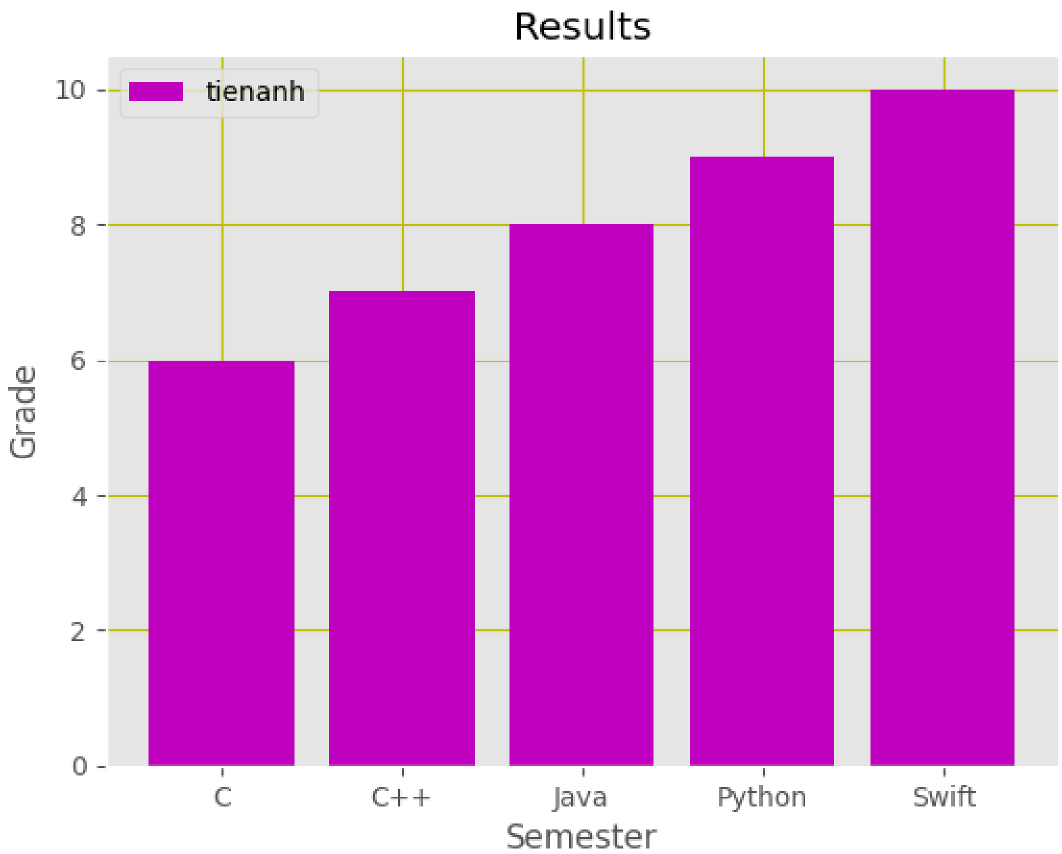
[19]:
```
plt.plot(
  subjects,
  tienanh_grade,
  label="tienanh",
  )
plt.plot(
  subjects,
  vphuong_grade,
  label="vphuong",
  )
plt.plot(
  subjects,
  vu_grade,
  label="vu",
  )
plt.plot(
  subjects,
  nam_grade,
  label="nam",
  )
plt.plot(
  subjects,
  dphuong_grade,
  label="dphuong",
  )

plt.title('Results')
plt.xlabel('Semester')
plt.ylabel('Grade')
```

[19]: Text(0, 0.5, 'Grade')



[20]:
```
style.use('ggplot')

subjects = subjects.tolist()
subjects

plt.bar(
  subjects,
  tienanh_grade,
  label="tienanh",
  color = "m",
  align = "center"
  )

plt.title('Results')
plt.xlabel('Semester')
plt.ylabel('Grade')
```

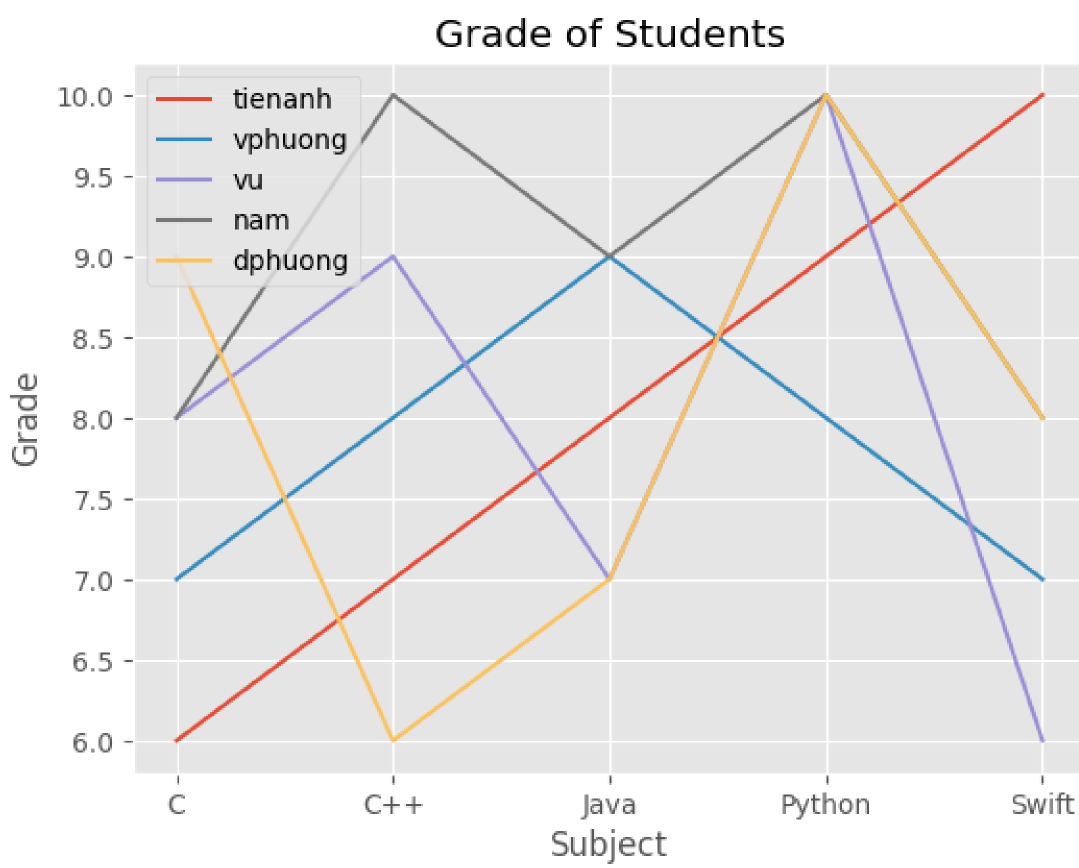
```
plt.legend()
plt.grid(True, color="y")
```



[21]:
```
plt.plot(
  subjects,
  tienanh_grade,
  label="tienanh",
  )
plt.plot(
  subjects,
  vphuong_grade,
  label="vphuong",
  )
plt.plot(
  subjects,
  vu_grade,
  label="vu",
  )
plt.plot(
```

```
  subjects,
  nam_grade,
  label="nam",
  )
plt.plot(
  subjects,
  dphuong_grade,
  label="dphuong",
  )

plt.xlabel("Subject")
plt.ylabel("Grade")
plt.title("Grade of Students")
plt.legend()
```

[21]: <matplotlib.legend.Legend at 0x1bfcf2cbeb0>



1.4 1.9

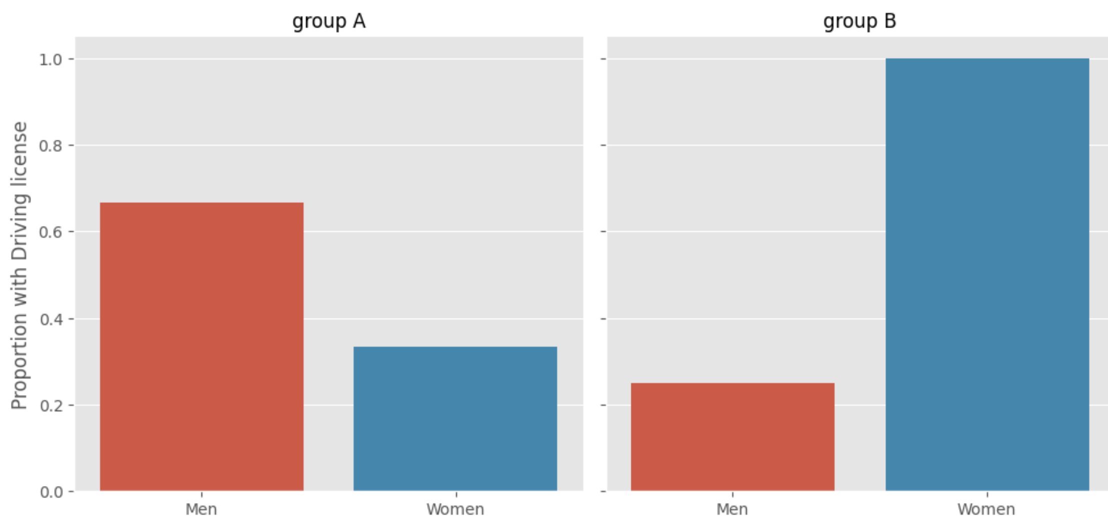
[22]:
```
df = pd.read_csv('data/no1_9.csv')
df
```

[22]:

	gender	group	license
0	men	A	1
1	men	A	0
2	men	A	1
3	women	A	1
4	women	A	0
5	women	A	0
6	men	B	0
7	men	B	0
8	men	B	0
9	men	B	1
10	women	B	1
11	women	B	1
12	women	B	1
13	women	B	1

[23]:
```
g = sns.catplot(x="gender", y="license", col="group",
                data = df, kind="bar", errorbar=None, aspect=1.0)

#--- set the labels ---
g.set_axis_labels("", "Proportion with Driving license")
g.set_xticklabels(["Men", "Women"])
g.set_titles("{col_var} {col_name}")

#--- show plot ---
plt.show()
```

```
c:\Users\tien2\miniconda3\lib\site-packages\seaborn\axisgrid.py:118:
UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



1.5 1.10

[24]:
```
df = pd.read_csv('data/no1_10.csv')
df
```

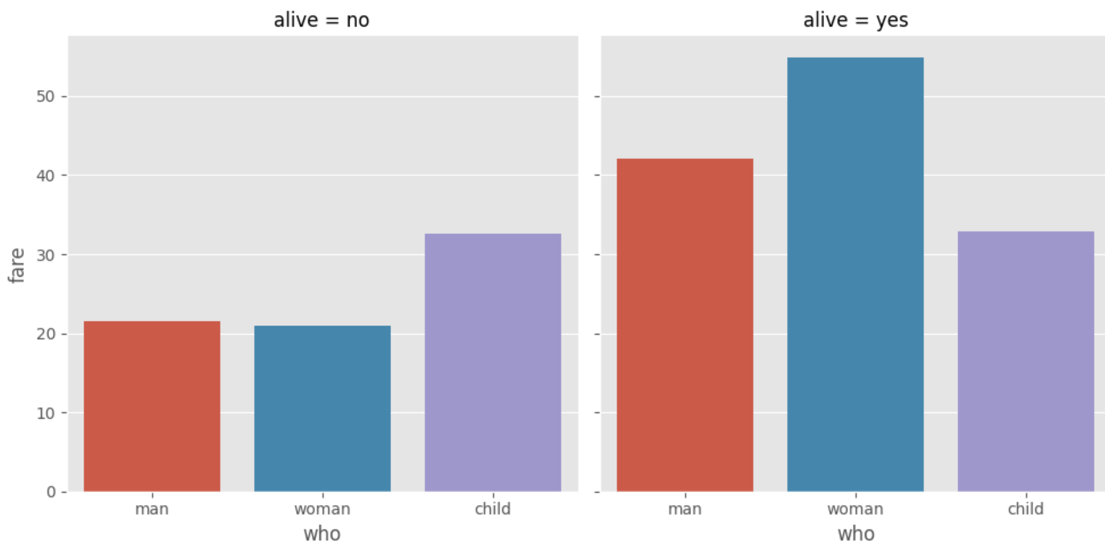
[24]:
```
     survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
0           0       3    male  22.0      1      0   7.2500        S   Third
1           1       1  female  38.0      1      0  71.2833        C   First
2           1       3  female  26.0      0      0   7.9250        S   Third
3           1       1  female  35.0      1      0  53.1000        S   First
4           0       3    male  35.0      0      0   8.0500        S   Third
..        ...     ...     ...   ...    ...    ...      ...      ...     ...
886         0       2    male  27.0      0      0  13.0000        S  Second
887         1       1  female  19.0      0      0  30.0000        S   First
888         0       3  female   NaN      1      2  23.4500        S   Third
889         1       1    male  26.0      0      0  30.0000        C   First
890         0       3    male  32.0      0      0   7.7500        Q   Third

       who  adult_male deck  embark_town alive  alone
0      man        True  NaN  Southampton    no  False
1    woman       False    C    Cherbourg   yes  False
2    woman       False  NaN  Southampton   yes   True
3    woman       False    C  Southampton   yes  False
4      man        True  NaN  Southampton    no   True
..     ...         ...  ...          ...   ...    ...
886    man        True  NaN  Southampton    no   True
887  woman       False    B  Southampton   yes   True
888  woman       False  NaN  Southampton    no  False
889    man        True    C    Cherbourg   yes   True
```

```
890     man         True  NaN   Queenstown    no   True

[891 rows x 15 columns]
```

[25]:

```python
g = sns.catplot(x="who", y="fare", col="alive",
                data=df, kind="bar", errorbar=None, aspect=1.0)
```

```
c:\Users\tien2\miniconda3\lib\site-packages\seaborn\axisgrid.py:118:
UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



1.6 1.11

[26]:

```python
sns.set_style("whitegrid")

#---load data---
data = pd.read_csv('data/salary.csv')

#---plot the swarm plot---
sns.swarmplot(x="gender", y="salary", data=data)

ax = plt.gca()
ax.set_title("Salary distribution")

#---show plot---
plt.show()
```

1.7 1.12

[27]:
```
data = np.array([(50, 2.5), (60, 3), (65, 3.5), (70, 3.8), (75, 4), (80, 4.5),␣
↪(85, 5)])
data
```

[27]:
```
array([[50. ,  2.5],
       [60. ,  3. ],
       [65. ,  3.5],
       [70. ,  3.8],
       [75. ,  4. ],
       [80. ,  4.5],
       [85. ,  5. ]])
```

[28]:
```
X = data[:,0].reshape(-1,1)
y = data[:,1]

model = LinearRegression()
model.fit(X, y)
```

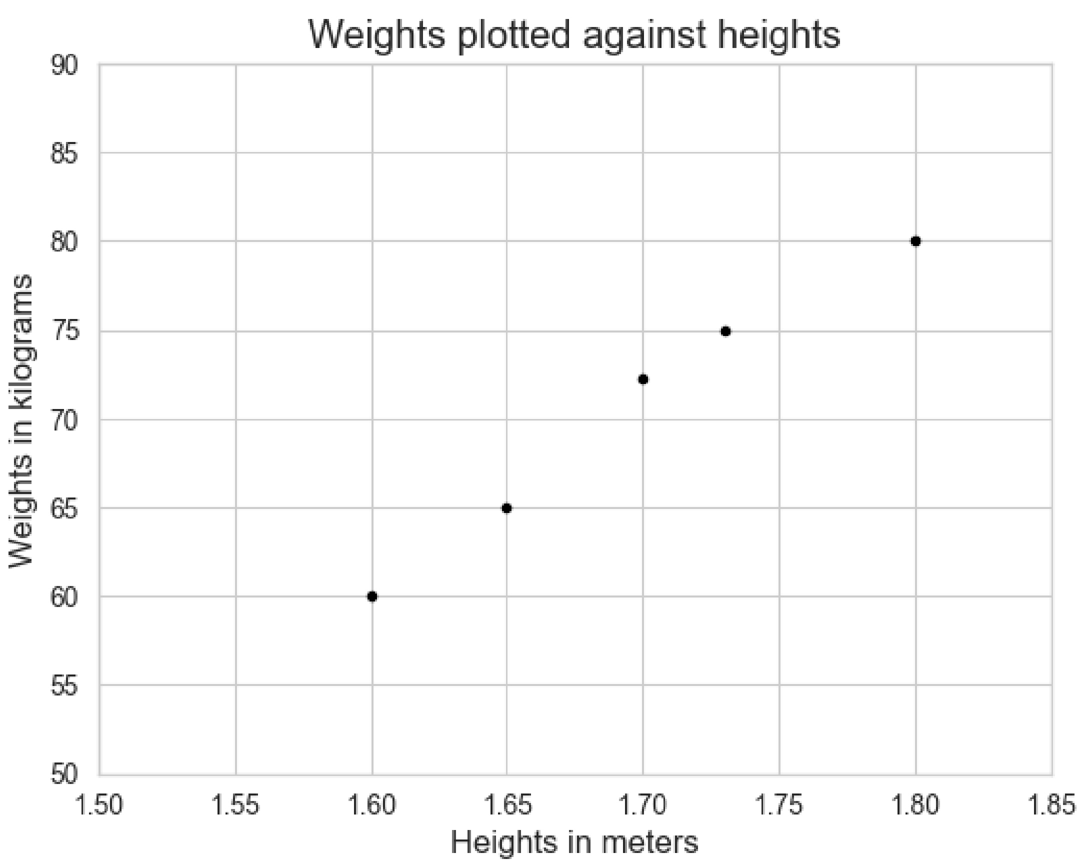
```
new_areas = np.array([55, 68, 76, 90]).reshape(-1, 1)
predicted_prices = model.predict(new_areas)

for area, price in zip(new_areas, predicted_prices):
    print(f"Din tích: {area} m2, Giá d đoán: {price:.2f} t")

plt.scatter(X, y, color='blue', label='Data')
plt.plot(X, model.predict(X), color='red', label='Linear Regression')
plt.scatter(new_areas, predicted_prices, color='green', label='Predictions')
plt.xlabel('Din tích (m2)')
plt.ylabel('Giá nhà (t)')
plt.title('D đoán giá nhà bng Linear Regression')
plt.legend()
plt.show()
```

```
Din tích: [55] m2, Giá d đoán: 2.75 t
Din tích: [68] m2, Giá d đoán: 3.67 t
Din tích: [76] m2, Giá d đoán: 4.23 t
Din tích: [90] m2, Giá d đoán: 5.21 t
```

1.8 1.13

[29]:
```
heights = [[1.6], [1.65], [1.7], [1.73], [1.8]]

weights = [[60], [65], [72.3], [75], [80]]
```

[30]:
```
# represents the weights of a group of people in kgs
weights = [[60], [65], [72.3], [75], [80]]

plt.title('Weights plotted against heights')
plt.xlabel('Heights in meters')
plt.ylabel('Weights in kilograms')

plt.plot(heights, weights, 'k.')

# axis range for x and y
plt.axis([1.5, 1.85, 50, 90])
plt.grid(True)
```

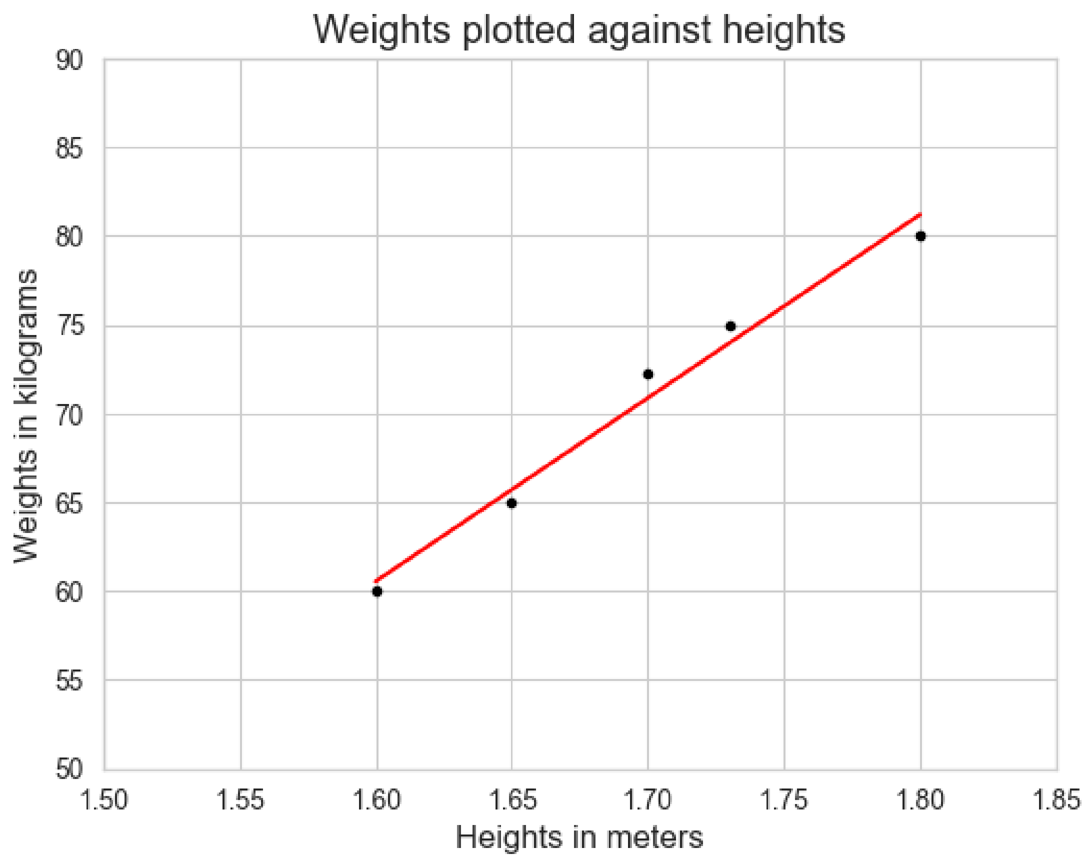


[31]:
```
model = LinearRegression()
model.fit(X=heights, y=weights)

weight = model.predict([[1.75]])[0][0]
print(f'Predicted weight for height 1.75 m: {round(weight,2)} kg')
```

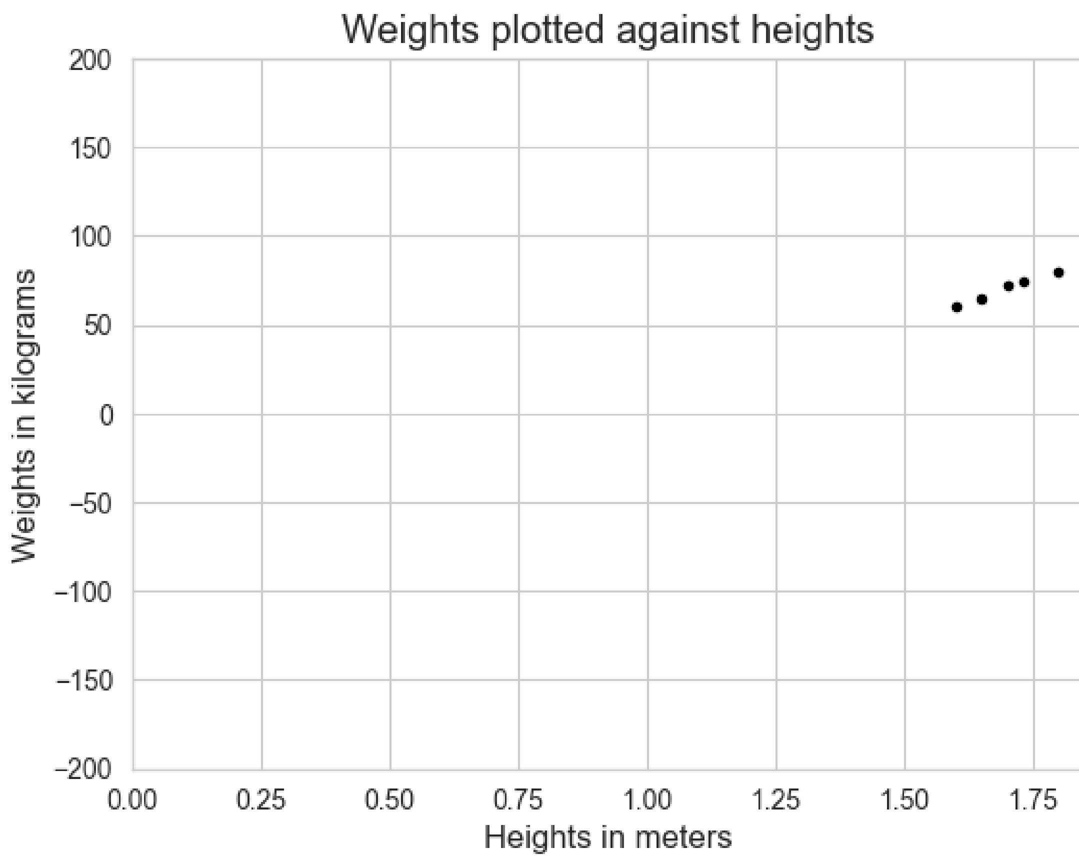
```
Predicted weight for height 1.75 m: 76.04 kg
```

[32]:
```
import matplotlib.pyplot as plt

heights = [[1.6], [1.65], [1.7], [1.73], [1.8]]
weights = [[60], [65], [72.3], [75], [80]]

plt.title('Weights plotted against heights')
plt.xlabel('Heights in meters')
plt.ylabel('Weights in kilograms')
plt.plot(heights, weights, 'k.')

plt.axis([1.5, 1.85, 50, 90])
plt.grid(True)

# plot the regression line
plt.plot(heights, model.predict(heights), color='r')

round(model.predict([[0]])[0][0],2) # -104.75

print(round(model.intercept_[0],2)) # -104.75

print(round(model.coef_[0][0],2)) # 103.31
```

```
-104.75
103.31
```



[33]:
```
plt.title('Weights plotted against heights')
plt.xlabel('Heights in meters')
plt.ylabel('Weights in kilograms')

plt.plot(heights, weights, 'k.')

plt.axis([0, 1.85, -200, 200])
plt.grid(True)
```



[34]:
```python
import numpy as np

print('Residual sum of squares: %.2f' %
       np.sum((weights - model.predict(heights)) ** 2))
```

```
Residual sum of squares: 5.34
```

[35]:
```python
# test data
heights_test = [[1.58], [1.62], [1.69], [1.76], [1.82]]
weights_test = [[58], [63], [72], [73], [85]]
```

[36]:
```python
# Total Sum of Squares (TSS)
weights_test_mean = np.mean(np.ravel(weights_test))
TSS = np.sum((np.ravel(weights_test) -
              weights_test_mean) ** 2)
print("TSS: %.2f" % TSS)

# Residual Sum of Squares (RSS)
RSS = np.sum((np.ravel(weights_test) -
              np.ravel(model.predict(heights_test)))
```

```
                    ** 2)
print("RSS: %.2f" % RSS)

# R_squared
R_squared = 1 - (RSS / TSS)
print("R-squared: %.2f" % R_squared)

# using scikit-learn to calculate r-squared
print('R-squared: %.4f' % model.score(heights_test,
                                      weights_test))
```

```
TSS: 430.80
RSS: 24.62
R-squared: 0.94
R-squared: 0.9429
```

[37]:
```
import pickle

# save the model to disk
filename = './data/HeightsAndWeights_model.sav'
# write to the file using write and binary mode
pickle.dump(model, open(filename, 'wb'))
```

[38]:
```
# load the model from disk
loaded_model = pickle.load(open(filename, 'rb'))
result = loaded_model.score(heights_test,
                            weights_test)
result
```

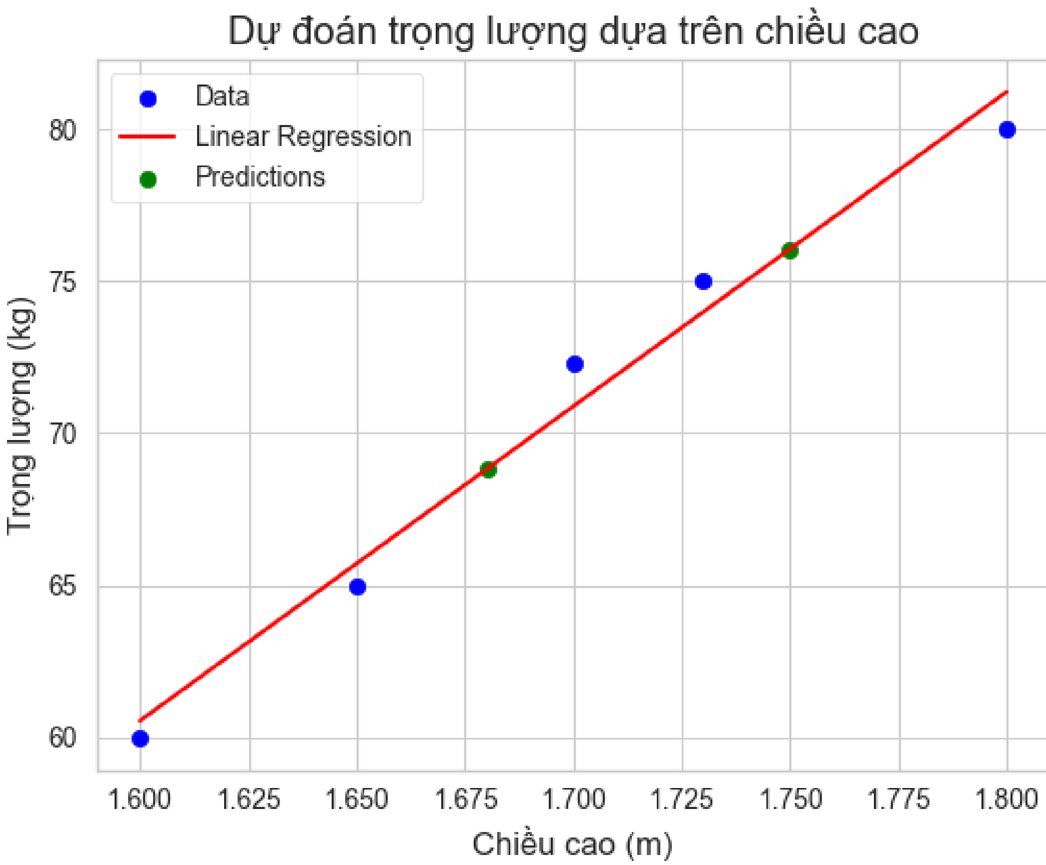
[38]: 0.9428592885995254

1.8.1 Personal records

[39]:
```
heights = [[1.6], [1.65], [1.7], [1.73], [1.8]]

weights = [[60], [65], [72.3], [75], [80]]

model = LinearRegression()
model.fit(heights, weights)

# Predict weights for new heights
new_heights = np.array([[1.68], [1.75]])
predicted_weights = model.predict(new_heights)

# Display predictions
for height, weight in zip(new_heights, predicted_weights):
    print(f"Chiu cao: {height[0]} m, Trng lng d đoán: {weight[0]:.2f} kg")
```

```
Chiu cao: 1.68 m, Trng lng d đoán: 68.81 kg
Chiu cao: 1.75 m, Trng lng d đoán: 76.04 kg
```

[40]:
```
# Visualization
plt.scatter(heights, weights, color='blue', label='Data')
plt.plot(heights, model.predict(heights), color='red', label='Linear Regression')
plt.scatter(new_heights, predicted_weights, color='green', label='Predictions')
plt.xlabel('Chiu cao (m)')
plt.ylabel('Trng lng (kg)')
plt.title('D đoán trng lng da trên chiu cao')
plt.legend()
plt.show()
```



1.9 1.14

[41]:
```
dataset = fetch_openml(name='boston')
dataset.data
```

```
c:\Users\tien2\miniconda3\lib\site-packages\sklearn\datasets\_openml.py:303:
UserWarning: Multiple active versions of the dataset matching the name boston
```

```
exist. Versions may be fundamentally different, returning version 1.
  warn(
c:\Users\tien2\miniconda3\lib\site-packages\sklearn\datasets\_openml.py:1002:
FutureWarning: The default value of `parser` will change from `'liac-arff'` to
`'auto'` in 1.4. You can set `parser='auto'` to silence this warning. Therefore,
an `ImportError` will be raised from 1.4 if the dataset is dense and pandas is
not installed. Note that the pandas parser may return different data types. See
the Notes Section in fetch_openml's API doc for details.
  warn(
```

[41]:

```
        CRIM    ZN  INDUS CHAS    NOX     RM   AGE     DIS RAD    TAX  \
0    0.00632  18.0   2.31    0  0.538  6.575  65.2  4.0900   1  296.0
1    0.02731   0.0   7.07    0  0.469  6.421  78.9  4.9671   2  242.0
2    0.02729   0.0   7.07    0  0.469  7.185  61.1  4.9671   2  242.0
3    0.03237   0.0   2.18    0  0.458  6.998  45.8  6.0622   3  222.0
4    0.06905   0.0   2.18    0  0.458  7.147  54.2  6.0622   3  222.0
..       ...   ...    ...  ...    ...    ...   ...     ...  ..    ...
501  0.06263   0.0  11.93    0  0.573  6.593  69.1  2.4786   1  273.0
502  0.04527   0.0  11.93    0  0.573  6.120  76.7  2.2875   1  273.0
503  0.06076   0.0  11.93    0  0.573  6.976  91.0  2.1675   1  273.0
504  0.10959   0.0  11.93    0  0.573  6.794  89.3  2.3889   1  273.0
505  0.04741   0.0  11.93    0  0.573  6.030  80.8  2.5050   1  273.0

     PTRATIO       B  LSTAT
0       15.3  396.90   4.98
1       17.8  396.90   9.14
2       17.8  392.83   4.03
3       18.7  394.63   2.94
4       18.7  396.90   5.33
..       ...     ...    ...
501     21.0  391.99   9.67
502     21.0  396.90   9.08
503     21.0  396.90   5.64
504     21.0  393.45   6.48
505     21.0  396.90   7.88

[506 rows x 13 columns]
```

[42]:

```python
dataset.feature_names
```

[42]:

```
['CRIM',
 'ZN',
 'INDUS',
 'CHAS',
 'NOX',
 'RM',
 'AGE',
 'DIS',
```

```
 'RAD',
 'TAX',
 'PTRATIO',
 'B',
 'LSTAT']
```

[43]:
```
dataset.DESCR
```

[43]:
```
"**Author**:   \n**Source**: Unknown - Date unknown  \n**Please cite**:
\n\nThe Boston house-price data of Harrison, D. and Rubinfeld, D.L.
'Hedonic\nprices and the demand for clean air', J. Environ. Economics &
Management,\nvol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression
diagnostics\n...', Wiley, 1980.   N.B. Various transformations are used in the
table on\npages 244-261 of the latter.\nVariables in order:\nCRIM     per capita
crime rate by town\nZN       proportion of residential land zoned for lots over
25,000 sq.ft.\nINDUS    proportion of non-retail business acres per town\nCHAS
Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)\nNOX
nitric oxides concentration (parts per 10 million)\nRM       average number of
rooms per dwelling\nAGE      proportion of owner-occupied units built prior to
1940\nDIS      weighted distances to five Boston employment centres\nRAD
index of accessibility to radial highways\nTAX      full-value property-tax rate
per $10,000\nPTRATIO  pupil-teacher ratio by town\nB        1000(Bk - 0.63)^2
where Bk is the proportion of blacks by town\nLSTAT    % lower status of the
population\nMEDV     Median value of owner-occupied homes in
$1000's\n\n\nInformation about the dataset\nCLASSTYPE: numeric\nCLASSINDEX:
last\n\nDownloaded from openml.org."
```

[44]:
```
dataset.target
```

[44]:
```
0      24.0
1      21.6
2      34.7
3      33.4
4      36.2
       ...
501    22.4
502    20.6
503    23.9
504    22.0
505    11.9
Name: MEDV, Length: 506, dtype: float64
```

[45]:
```
df = pd.DataFrame(dataset.data, columns=dataset.feature_names)
df.head()
```

[45]:
```
      CRIM    ZN  INDUS CHAS    NOX     RM   AGE     DIS RAD    TAX  PTRATIO  \
0  0.00632  18.0   2.31    0  0.538  6.575  65.2  4.0900   1  296.0     15.3
```

```
1  0.02731   0.0   7.07    0  0.469  6.421  78.9  4.9671    2  242.0     17.8
2  0.02729   0.0   7.07    0  0.469  7.185  61.1  4.9671    2  242.0     17.8
3  0.03237   0.0   2.18    0  0.458  6.998  45.8  6.0622    3  222.0     18.7
4  0.06905   0.0   2.18    0  0.458  7.147  54.2  6.0622    3  222.0     18.7

        B  LSTAT
0  396.90   4.98
1  396.90   9.14
2  392.83   4.03
3  394.63   2.94
4  396.90   5.33
```

[46]:
```
df['MEDV']=dataset.target
df.head()
```

[46]:
```
      CRIM    ZN  INDUS CHAS    NOX     RM   AGE     DIS RAD    TAX  PTRATIO  \
0  0.00632  18.0   2.31    0  0.538  6.575  65.2  4.0900   1  296.0     15.3
1  0.02731   0.0   7.07    0  0.469  6.421  78.9  4.9671   2  242.0     17.8
2  0.02729   0.0   7.07    0  0.469  7.185  61.1  4.9671   2  242.0     17.8
3  0.03237   0.0   2.18    0  0.458  6.998  45.8  6.0622   3  222.0     18.7
4  0.06905   0.0   2.18    0  0.458  7.147  54.2  6.0622   3  222.0     18.7

        B  LSTAT  MEDV
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
4  396.90   5.33  36.2
```

[47]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    category
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    category
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
```

```
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: category(2), float64(12)
memory usage: 49.0 KB
```

[48]:
```
print(df.isnull().sum())
```

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

[49]:
```
corr = df.corr()
corr
```

[49]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE
CRIM	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734
ZN	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537
INDUS	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779
CHAS	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518
NOX	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470
RM	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265
AGE	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000
DIS	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881
RAD	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022
TAX	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456
PTRATIO	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515
B	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534
LSTAT	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339
MEDV	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955

	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
CRIM	-0.379670	0.625505	0.582764	0.289946	-0.385064	0.455621	-0.388305
ZN	0.664408	-0.311948	-0.314563	-0.391679	0.175520	-0.412995	0.360445
INDUS	-0.708027	0.595129	0.720760	0.383248	-0.356977	0.603800	-0.483725
CHAS	-0.099176	-0.007368	-0.035587	-0.121515	0.048788	-0.053929	0.175260
NOX	-0.769230	0.611441	0.668023	0.188933	-0.380051	0.590879	-0.427321

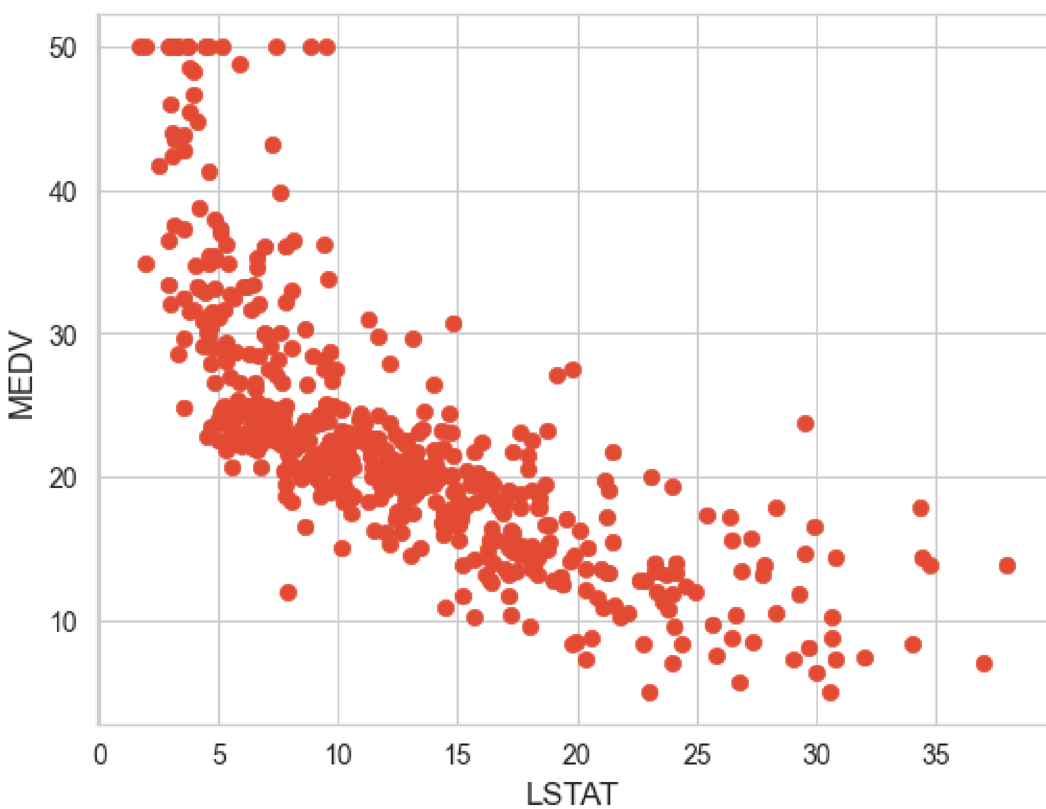
```
RM       0.205246 -0.209847 -0.292048 -0.355501  0.128069 -0.613808  0.695360
AGE     -0.747881  0.456022  0.506456  0.261515 -0.273534  0.602339 -0.376955
DIS      1.000000 -0.494588 -0.534432 -0.232471  0.291512 -0.496996  0.249929
RAD     -0.494588  1.000000  0.910228  0.464741 -0.444413  0.488676 -0.381626
TAX     -0.534432  0.910228  1.000000  0.460853 -0.441808  0.543993 -0.468536
PTRATIO -0.232471  0.464741  0.460853  1.000000 -0.177383  0.374044 -0.507787
B        0.291512 -0.444413 -0.441808 -0.177383  1.000000 -0.366087  0.333461
LSTAT   -0.496996  0.488676  0.543993  0.374044 -0.366087  1.000000 -0.737663
MEDV     0.249929 -0.381626 -0.468536 -0.507787  0.333461 -0.737663  1.000000
```

[50]:
```
print(corr.abs().nlargest(3, 'MEDV').index)

print(corr.abs().nlargest(3, 'MEDV').values[:,13])
```

```
Index(['MEDV', 'LSTAT', 'RM'], dtype='object')
[1.         0.73766273 0.69535995]
```

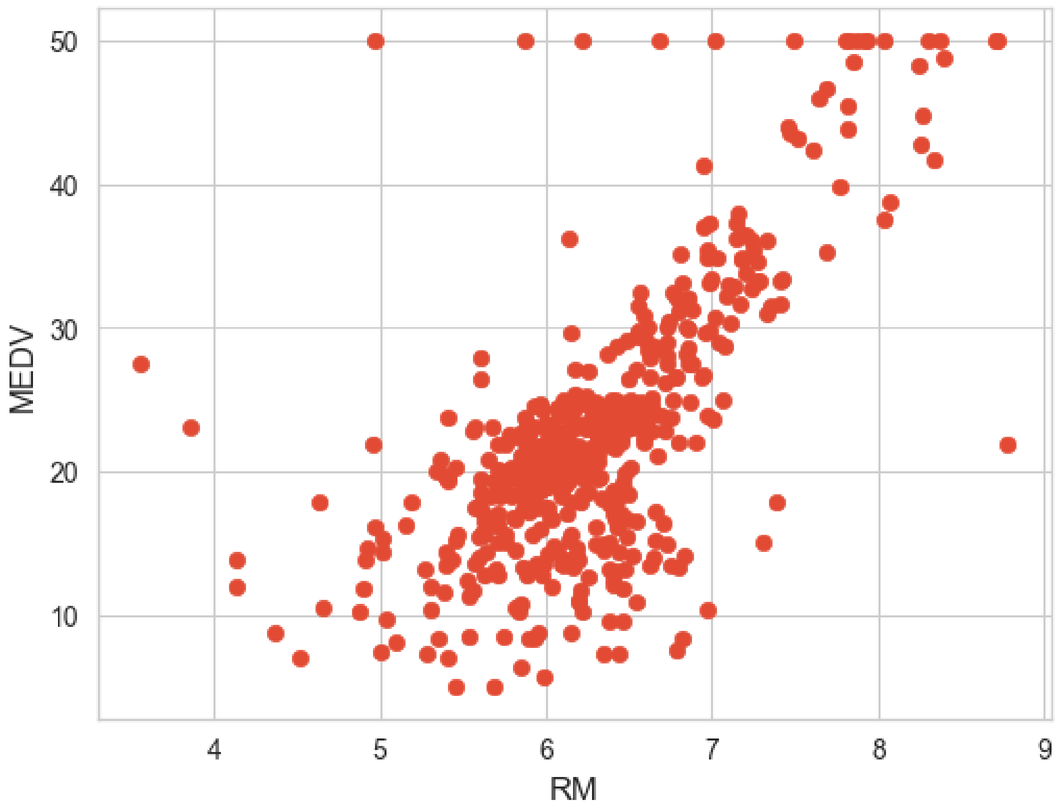
[51]:
```
plt.scatter(df['LSTAT'], df['MEDV'], marker='o')
plt.xlabel('LSTAT')
plt.ylabel('MEDV')
```

[51]: `Text(0, 0.5, 'MEDV')`

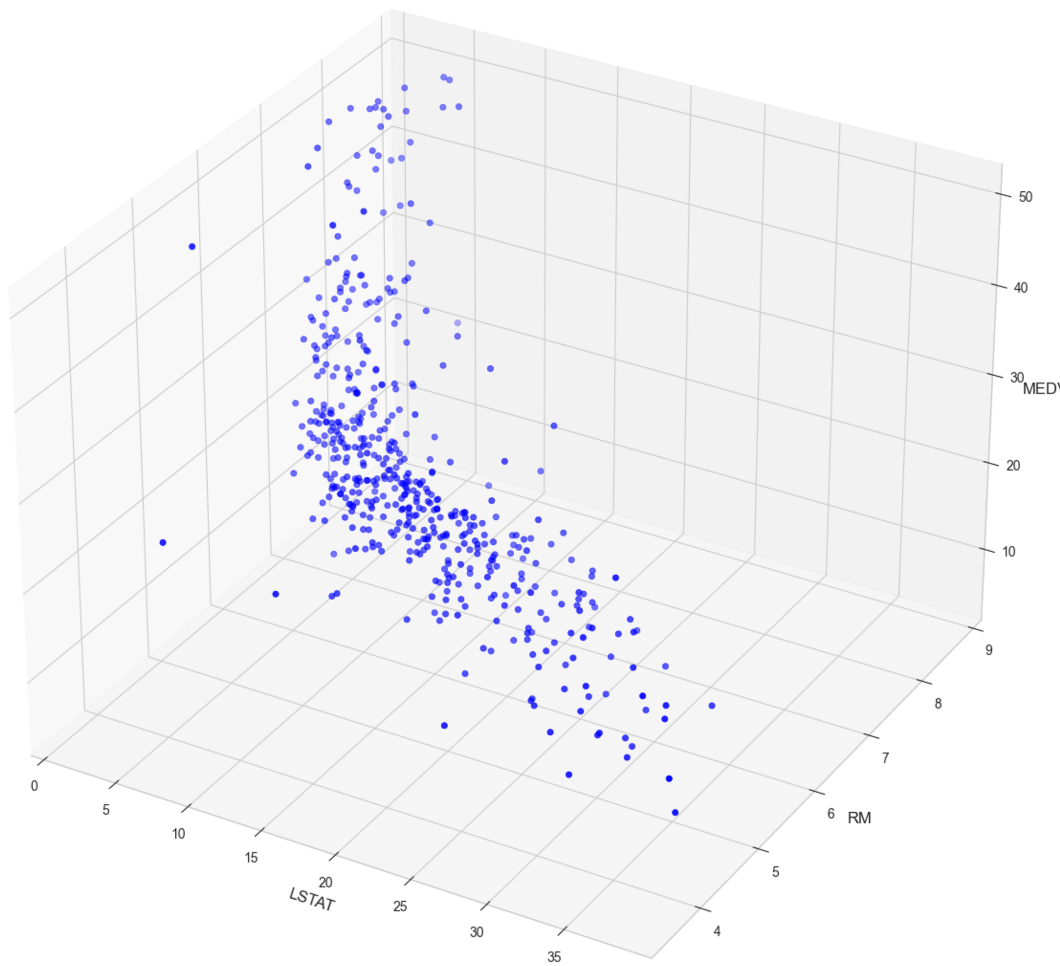


[52]:
```
plt.scatter(df['RM'], df['MEDV'], marker='o')
plt.xlabel('RM')
plt.ylabel('MEDV')
```

[52]: Text(0, 0.5, 'MEDV')



[53]:
```
fig = plt.figure(figsize=(18,15))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(df['LSTAT'],
           df['RM'],
           df['MEDV'],
           c='b')

ax.set_xlabel("LSTAT")
ax.set_ylabel("RM")
ax.set_zlabel("MEDV")
plt.show()
```



[54]:
```
x = pd.DataFrame(np.c_[df['LSTAT'], df['RM']], columns = ['LSTAT','RM'])
Y = df['MEDV']
```

[55]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, Y_train, Y_test = train_test_split(x, Y, test_size = 0.3,␣
 ↪random_state=5)
```

[56]:
```
print(x_train.shape)
print(Y_train.shape)
```

```
(354, 2)
(354,)
```

[57]:
```
print(x_test.shape)
print(Y_test.shape)
```

```
(152, 2)
(152,)
```

[58]:
```
model = LinearRegression()
model.fit(x_train, Y_train)
price_prediction = model.predict(x_test)
```

[59]:
```
print('R-Squared: %.4f' % model.score(x_test,Y_test))
```

```
R-Squared: 0.6162
```

[60]:
```
mse = mean_squared_error(Y_test, price_prediction)
mse
```

[60]: 36.49422110915324

[61]:
```
plt.scatter(Y_test, price_prediction)
plt.xlabel("Actual price")
plt.ylabel("Predicted prices")
plt.title("Actual prices vs Predicted prices")
```

[61]: Text(0.5, 1.0, 'Actual prices vs Predicted prices')

[62]:
```
print(model.intercept_)
print(model.coef_)
```

```
0.38437936780346504
[-0.65957972  4.83197581]
```

[63]:
```
print(model.predict([[30,5]]))
```

```
[4.75686695]
```

```
c:\Users\tien2\miniconda3\lib\site-packages\sklearn\base.py:464: UserWarning: X
does not have valid feature names, but LinearRegression was fitted with feature
names
  warnings.warn(
```

1.9.1 Plotting the 3D Hyperlane

[64]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
```

```python
from sklearn.datasets import fetch_openml

dataset = fetch_openml(name='boston')

df = pd.DataFrame(dataset.data, columns=dataset.feature_names)
df['MEDV'] = dataset.target

x = pd.DataFrame(np.c_[df['LSTAT'], df['RM']], columns = ['LSTAT','RM'])
Y = df['MEDV']

fig = plt.figure(figsize=(18,15))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(x['LSTAT'],
           x['RM'],
           Y,
           c='b')

ax.set_xlabel("LSTAT")
ax.set_ylabel("RM")
ax.set_zlabel("MEDV")

#---create a meshgrid of all the values for LSTAT and RM---
x_surf = np.arange(0, 40, 1)   #---for LSTAT---
y_surf = np.arange(0, 10, 1)   #---for RM---
x_surf, y_surf = np.meshgrid(x_surf, y_surf)

from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, Y)

#---calculate z(MEDC) based on the model---
z = lambda x,y: (model.intercept_ + model.coef_[0] * x + model.coef_[1] * y)

ax.plot_surface(x_surf, y_surf, z(x_surf,y_surf),
                rstride=1,
                cstride=1,
                color='None',
                alpha = 0.4)

plt.show()
```

```
c:\Users\tien2\miniconda3\lib\site-packages\sklearn\datasets\_openml.py:303:
UserWarning: Multiple active versions of the dataset matching the name boston
exist. Versions may be fundamentally different, returning version 1.
  warn(
c:\Users\tien2\miniconda3\lib\site-packages\sklearn\datasets\_openml.py:1002:
```

```
FutureWarning: The default value of `parser` will change from `'liac-arff'` to
`'auto'` in 1.4. You can set `parser='auto'` to silence this warning. Therefore,
an `ImportError` will be raised from 1.4 if the dataset is dense and pandas is
not installed. Note that the pandas parser may return different data types. See
the Notes Section in fetch_openml's API doc for details.
  warn(
```