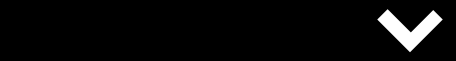
ITP 449

Regression

Lecture 8



Regression:

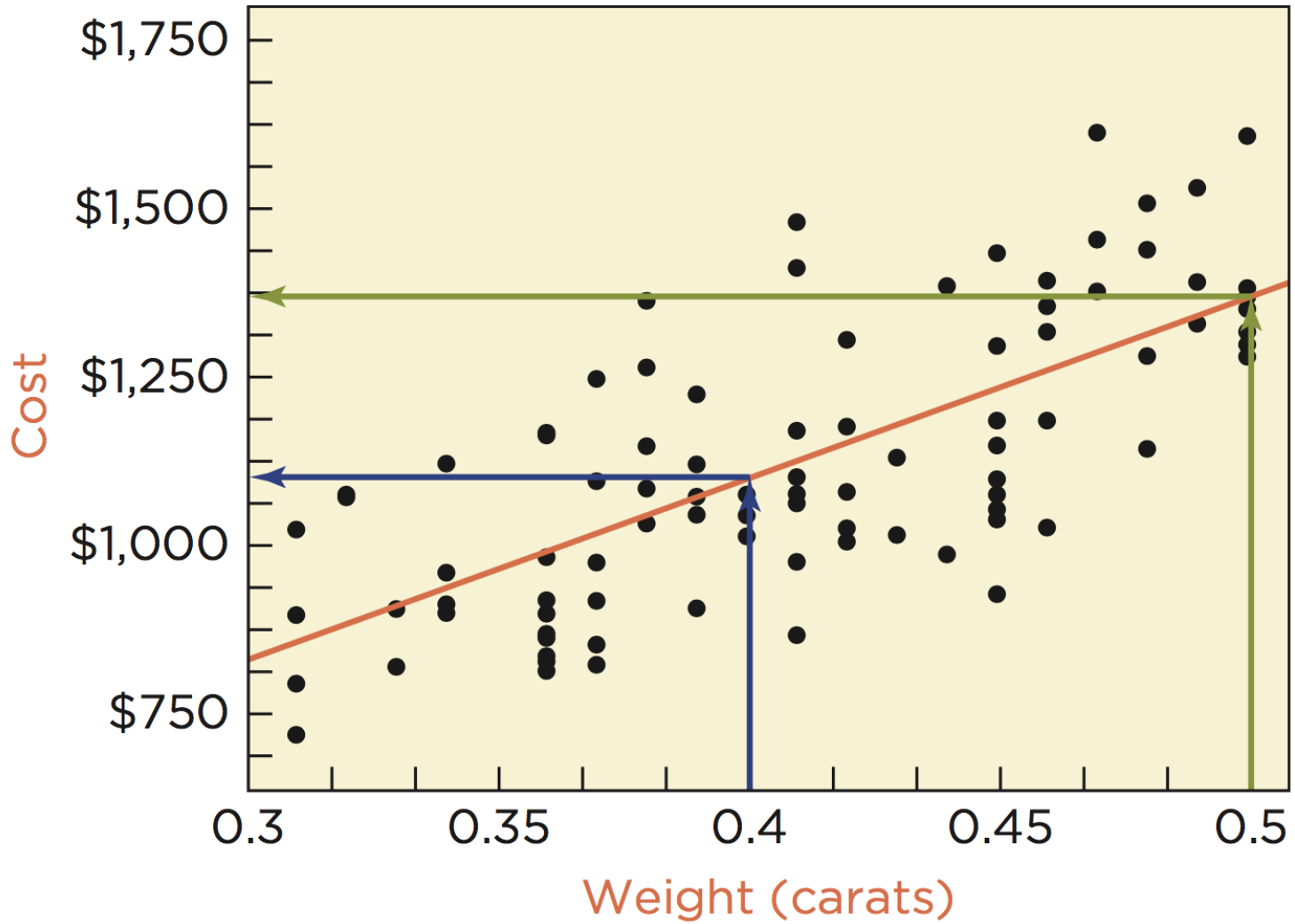
- Linear
- Logistic

Linear Regression

Equation of a Line

Identify the line fit to the data by an intercept b_0 and a slope b_1

The equation of the line is $\hat{y} = b_0 + b_1 x$

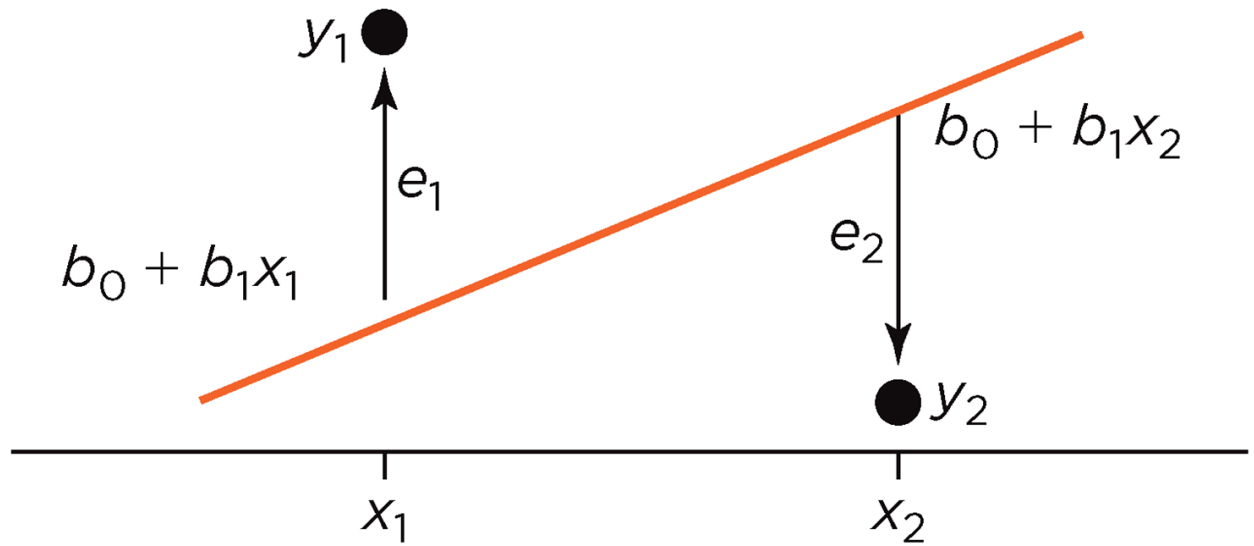

$1,750
$1,500
$1,250
$1,000
$750
Cost
0.3
0.35
0.4
0.45
0.5
Weight (carats)

Least Squares

Residual Vertical deviation of a point from the line

The best fitting line collectively makes the squares of the residuals as small as possible

The choices of b_0 and b_1 minimize the sum of the squared residuals


y_1
e_1
$b_0 + b_1x_1$
$b_0 + b_1x_2$
e_2
y_2
x_1
x_2

Interpretation

Intercept Average response when x = 0 and where the line crosses the *y* axis

Slope Estimates the marginal cost used to find the variable cost

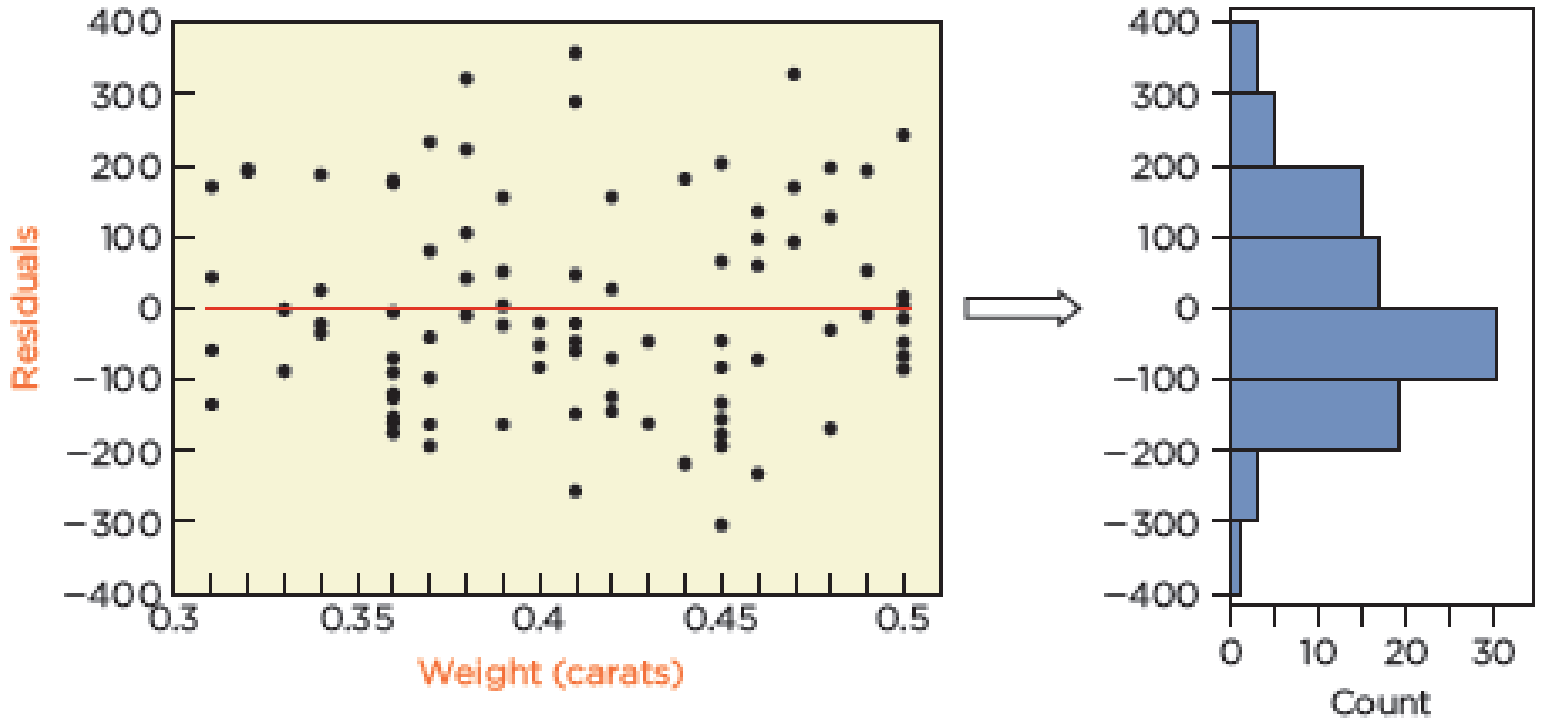
Residuals

Show variation that remains in the data after accounting for the linear relationship defined by the fitted line

If the least squares line captures the association between x and y, then a plot of residuals versus x should have no pattern

Standard Deviation of Residuals

Measures how much the residuals vary around the fitted line



R-Squared (r^2)

Square of the correlation between *x* and *y*

Fraction of the variation accounted for by the least squares regression line

Regression Checklist

No obvious lurking variables:

Are there other explanatory variables?

Linear:

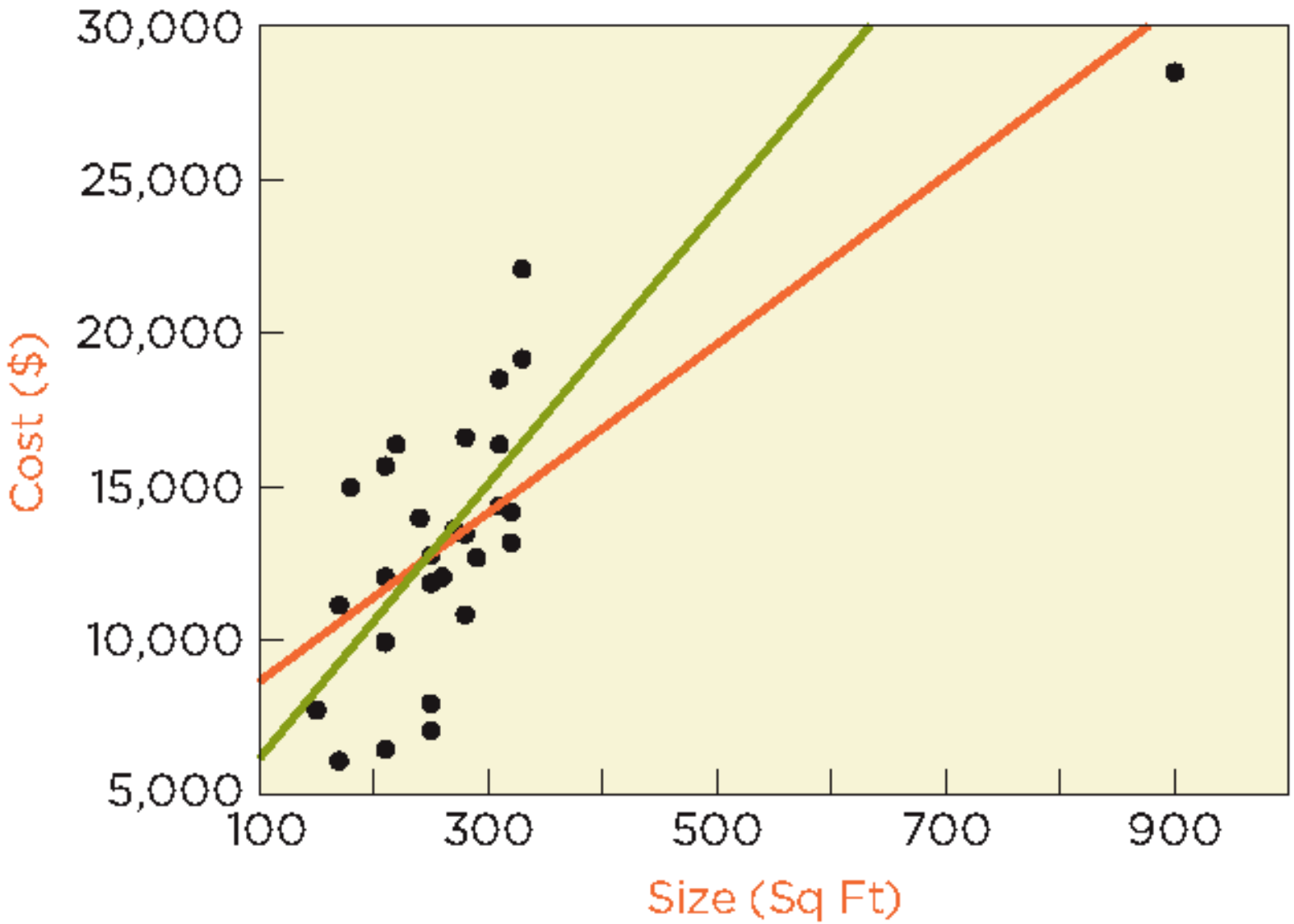
Use scatterplots

Random residual variation:

Use residual plots to make sure there is no pattern

Outliers

A contractor is bidding on a project to construct an 875 square foot addition to a home. How much does he bid?


30,000
25,000
20,000
15,000
10,000
5,000
Cost ($)
100
300
500
700
900
Size (Sq Ft)

https://www.kaggle.com/c/boston-housing/data

kaggle Search Competitions Datasets Kernels Discussion Learn ...

Boston Housing

Boston Housing

75 teams · 2 years ago

Overview Data Discussion Leaderboard Rules Late Submission

Data Description

Data (14 KB) API `kaggle competitions download -c boston-housing` ? Download All

Data Sources

- submission_example.csv 173 x 2
- test.csv 173 x 14
- train.csv 333 x 15

About this file

No description yet

Columns

- ID
- # crim
- # zn
- # indus
- # chas
- # nox
- # rm
- # age
- # dis
- # rad
- # tax
- # ptratio
- # black
- # lstat
- # medv

crim
per capita crime rate by town.

zn
proportion of residential land zoned for lots over 25,000 sq.ft.

indus
proportion of non-retail business acres per town.

chas
Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).

nox
nitrogen oxides concentration (parts per 10 million).

rm
average number of rooms per dwelling.

age
proportion of owner-occupied units built prior to 1940.

dis
weighted mean of distances to five Boston employment centres.

rad
index of accessibility to radial highways.

tax
full-value property-tax rate per \$10,000.

ptratio
pupil-teacher ratio by town.

black
1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town.

lstat
lower status of the population (percent).

medv
median value of owner-occupied homes in \$1000s.

Do the following:

Create a DataFrame variable containing the Boston Housing training data CSV file.

File Edit View Navigate Code Refactor Run Tools VCS Window Help ITP449_Fall2020 - in_class_coding.py

ITP449_Fall2020 〉 Class 〉 In Class Coding 〉 in_class_coding.py

in_class_coding

Pr...

in_class_coding.py

```
ITP449_Fall2020 C
  Class
    .idea
    Files
    In Class Codi
      in_class_c
    venv library
  External Libraries
  Scratches and Cons
```

```
1  import pandas as pd
2  import os
3
4  os.chdir('C:/Users/Reza/Desktop/ITP449_Fall2020/Class/Files')
5  pd.set_option('display.max_columns', None)
6
7  boston = pd.read_csv('BostonHousing.csv')
8  print(boston.head())
9  print(boston.describe())
10
```

10 183

Run: in_class_coding

```
      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     15.3
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242     17.8
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242     17.8
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222     18.7
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222     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
```

4: Run TODO 6: Problems Terminal Python Console Event Log

12:1 CRLF UTF-8 4 spaces Python 3.8 (Class)

1: Project 7: Structure 2: Favorites

```
C:\Users\Reza\Desktop\ITP449_Fall2020\Class\venv\Scripts\python.exe "C:/Users/Reza/Desktop/ITP449_Fall2020/Class/In Class Coding/in_class_coding.py"
              CRIM          ZN       INDUS        CHAS         NOX          RM  \
count   506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean      3.613524   11.363636   11.136779    0.069170    0.554695    6.284634
std       8.601545   23.322453    6.860353    0.253994    0.115878    0.702617
min       0.006320    0.000000    0.460000    0.000000    0.385000    3.561000
25%       0.082045    0.000000    5.190000    0.000000    0.449000    5.885500
50%       0.256510    0.000000    9.690000    0.000000    0.538000    6.208500
75%       3.677082   12.500000   18.100000    0.000000    0.624000    6.623500
max      88.976200  100.000000   27.740000    1.000000    0.871000    8.780000

               AGE         DIS         RAD         TAX     PTRATIO           B  \
count   506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     68.574901    3.795043    9.549407  408.237154   18.455534  356.674032
std      28.148861    2.105710    8.707259  168.537116    2.164946   91.294864
min       2.900000    1.129600    1.000000  187.000000   12.600000    0.320000
25%      45.025000    2.100175    4.000000  279.000000   17.400000  375.377500
50%      77.500000    3.207450    5.000000  330.000000   19.050000  391.440000
75%      94.075000    5.188425   24.000000  666.000000   20.200000  396.225000
max     100.000000   12.126500   24.000000  711.000000   22.000000  396.900000

            LSTAT        MEDV
count  506.000000  506.000000
mean    12.653063   22.532806
std      7.141062    9.197104
min      1.730000    5.000000
25%      6.950000   17.025000
50%     11.360000   21.200000
75%     16.955000   25.000000
max     37.970000   50.000000
```

Do the following:

Determine what the dependent variable (target array) is.

	CRIM	ZN	INDUS	CHAS	NOX	RM	\
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	
75%	3.677082	12.500000	18.100000	0.000000	0.624000	6.623500	
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	

	AGE	DIS	RAD	TAX	PTRATIO	B	\
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	
mean	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	
std	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	
min	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	
25%	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	
50%	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	
75%	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	
max	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	

	LSTAT	MEDV
count	506.000000	506.000000
mean	12.653063	22.532806
std	7.141062	9.197104
min	1.730000	5.000000
25%	6.950000	17.025000
50%	11.360000	21.200000
75%	16.955000	25.000000
max	37.970000	50.000000

median value of owner-occupied homes in $1000s

Do the following:

Cleanup the DataFrame by removing non-numeric columns and discrete numeric columns

1: Project

Pr...

ITP449_Fall2020 C
Class
.idea
Files
In Class Codi
in_class_c
venv library
External Libraries
Scratches and Cons

in_class_coding.py

```
1  import pandas as pd
2  import os
3
4  os.chdir('C:/Users/Reza/Desktop/ITP449_Fall2020/Class/Files')
5  pd.set_option('display.max_columns', None)
6
7  boston = pd.read_csv('BostonHousing.csv')
8
9  bostonNum = boston.drop(['CHAS'], axis=1)
10 print(bostonNum.dtypes)
11 print(bostonNum)
12
```

10 183

Run: in_class_coding

```
CRIM      float64
ZN        float64
INDUS     float64
NOX       float64
RM        float64
AGE       float64
```

Do the following:

Convert all columns in the DataFrame to float type

Pr...

ITP449_Fall2020 C
Class
.idea
Files
In Class Codi
in_class_c
venv library
External Libraries
Scratches and Cons

in_class_coding.py

```python
import pandas as pd
import os

os.chdir('C:/Users/Reza/Desktop/ITP449_Fall2020/Class/Files')
pd.set_option('display.max_columns', None)

boston = pd.read_csv('BostonHousing.csv')

bostonNum = boston.drop(['CHAS'], axis=1)

bostonNum['RAD'] = bostonNum['RAD'].astype(float)
bostonNum['TAX'] = bostonNum['TAX'].astype(float)

print(bostonNum.dtypes)
print(bostonNum)
```

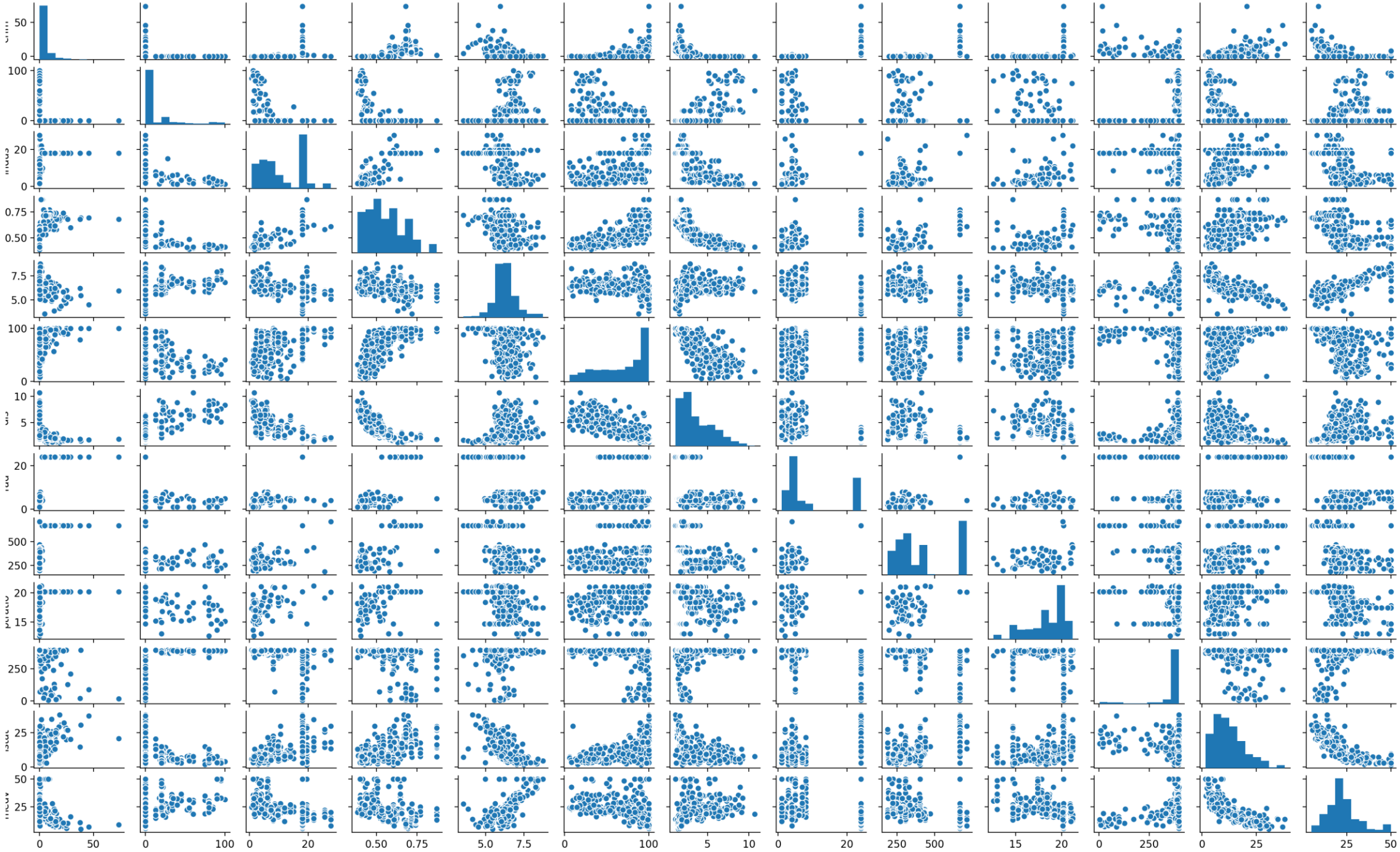
10 183

Run: in_class_coding

```
INDUS      float64
NOX        float64
```

Do the following:

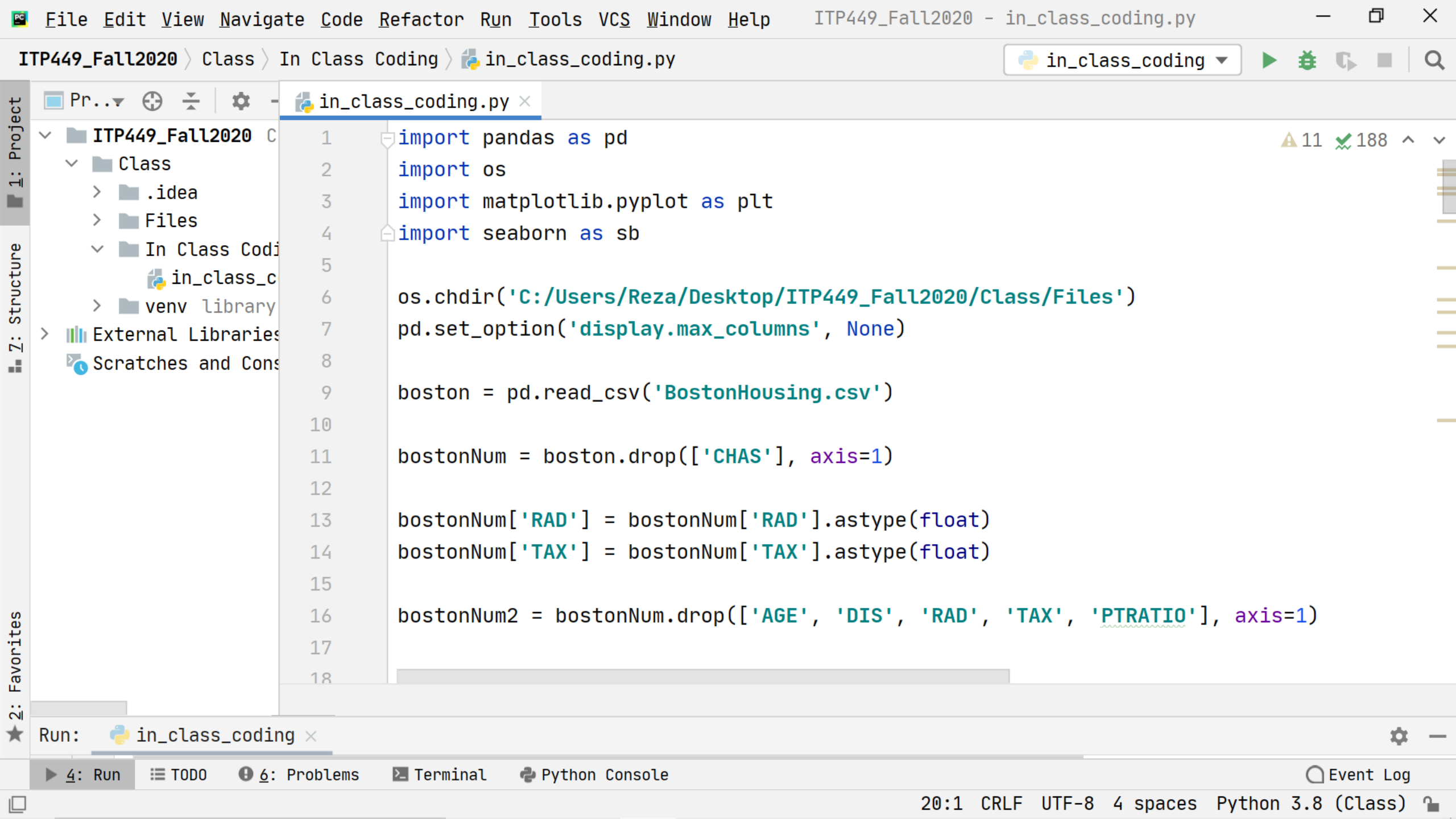
Create a scatterplot matrix to get an overview of the relationships between all the variable pairs.



```python
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sb

os.chdir('C:/Users/Reza/Desktop/ITP449_Fall2020/Class/Files')
pd.set_option('display.max_columns', None)

boston = pd.read_csv('BostonHousing.csv')

bostonNum = boston.drop(['CHAS'], axis=1)

bostonNum['RAD'] = bostonNum['RAD'].astype(float)
bostonNum['TAX'] = bostonNum['TAX'].astype(float)

sb.pairplot(bostonNum)
plt.show()
```

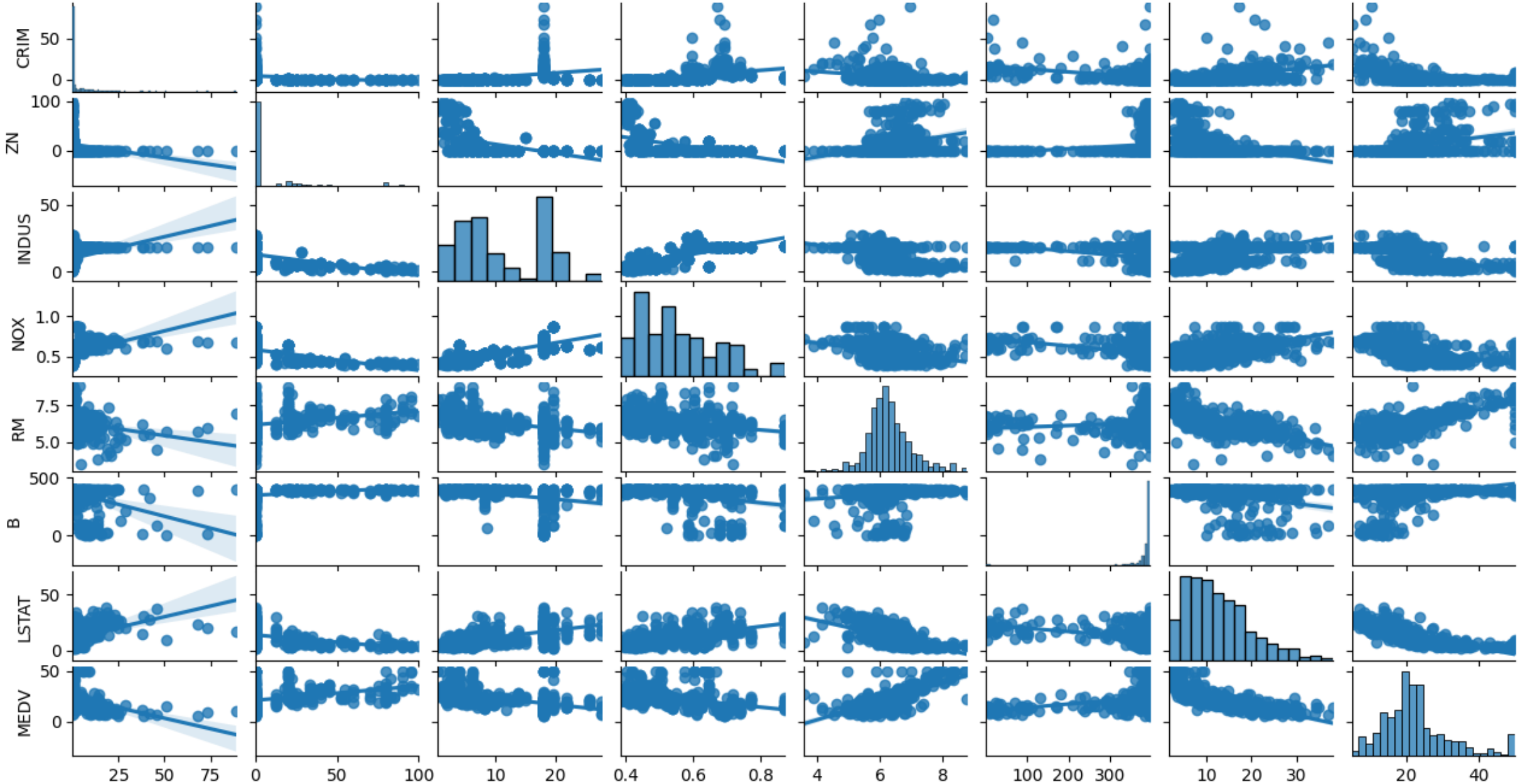
Do the following:

From the scatterplot matrix, determine which features have a strong relationship with the target. Update your cleaned up DataFrame to only include those columns.


File Edit View Navigate Code Refactor Run Tools VCS Window Help
ITP449_Fall2020 - in_class_coding.py
ITP449_Fall2020 〉 Class 〉 In Class Coding 〉 in_class_coding.py
in_class_coding
1: Project
7: Structure
2: Favorites
Pr...
in_class_coding.py
ITP449_Fall2020 C
Class
.idea
Files
In Class Codi
in_class_c
venv library
External Libraries
Scratches and Cons
11
188
1 import pandas as pd
2 import os
3 import matplotlib.pyplot as plt
4 import seaborn as sb
5
6 os.chdir('C:/Users/Reza/Desktop/ITP449_Fall2020/Class/Files')
7 pd.set_option('display.max_columns', None)
8
9 boston = pd.read_csv('BostonHousing.csv')
10
11 bostonNum = boston.drop(['CHAS'], axis=1)
12
13 bostonNum['RAD'] = bostonNum['RAD'].astype(float)
14 bostonNum['TAX'] = bostonNum['TAX'].astype(float)
15
16 bostonNum2 = bostonNum.drop(['AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO'], axis=1)
17
18
Run: in_class_coding
4: Run
TODO
6: Problems
Terminal
Python Console
Event Log
20:1 CRLF UTF-8 4 spaces Python 3.8 (Class)

Do the following:

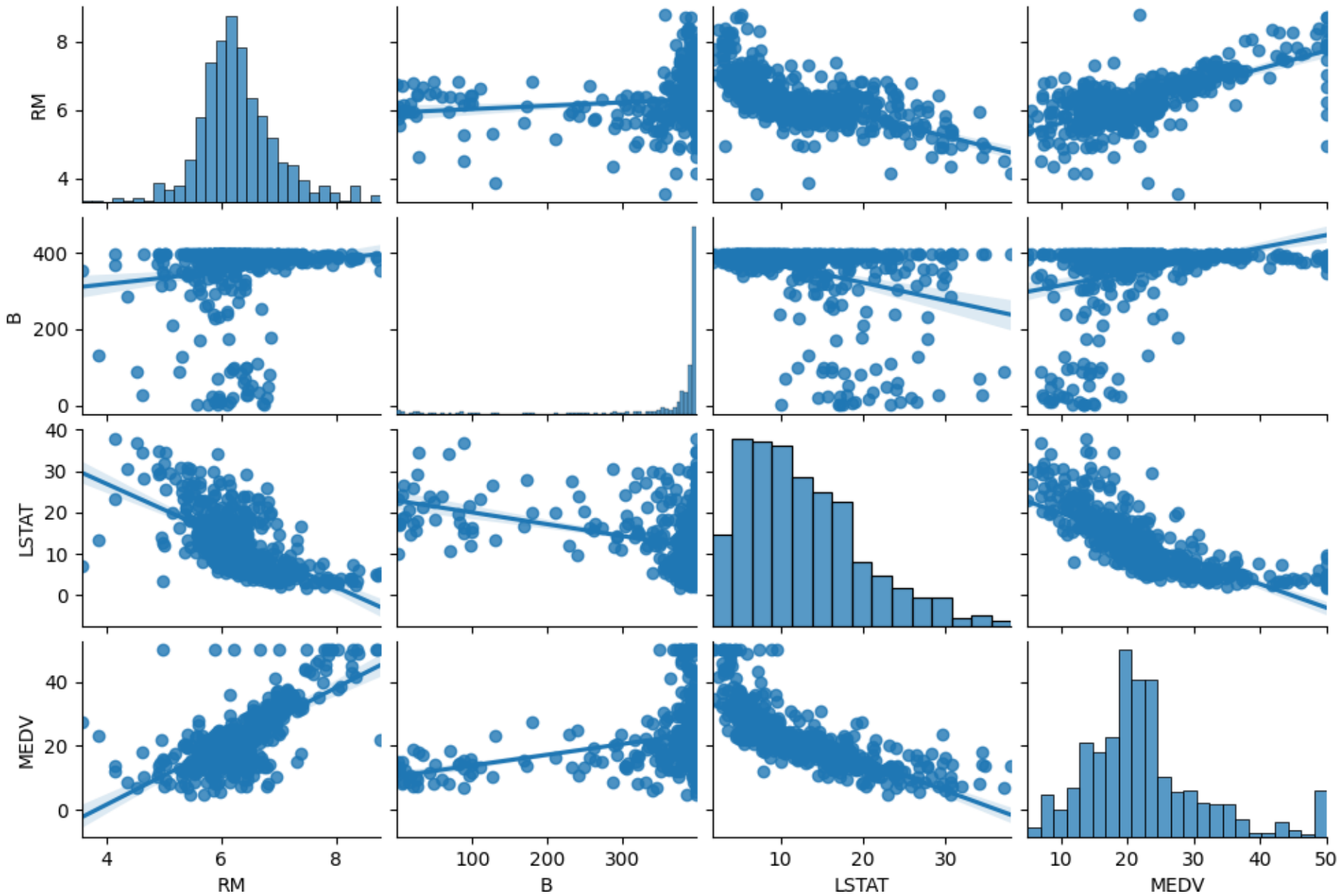
Redo the scatterplot matrix with the updated features. Include a regression line.


CRIM
ZN
INDUS
NOX
RM
B
LSTAT
MEDV
50
0
100
50
0
50
0
1.0
0.5
7.5
5.0
500
0
50
0
50
0
25
50
75
0
50
100
10
20
0.4
0.6
0.8
4
6
8
100
200
300
10
20
30
20
40

```python
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sb

os.chdir('C:/Users/Reza/Desktop/ITP449_Fall2020/Class/Files')
pd.set_option('display.max_columns', None)

boston = pd.read_csv('BostonHousing.csv')

bostonNum = boston.drop(['CHAS'], axis=1)

bostonNum['RAD'] = bostonNum['RAD'].astype(float)
bostonNum['TAX'] = bostonNum['TAX'].astype(float)

bostonNum2 = bostonNum.drop(['AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO'], axis=1)

sb.pairplot(bostonNum2, kind='reg')
plt.show()
```

Do the following:

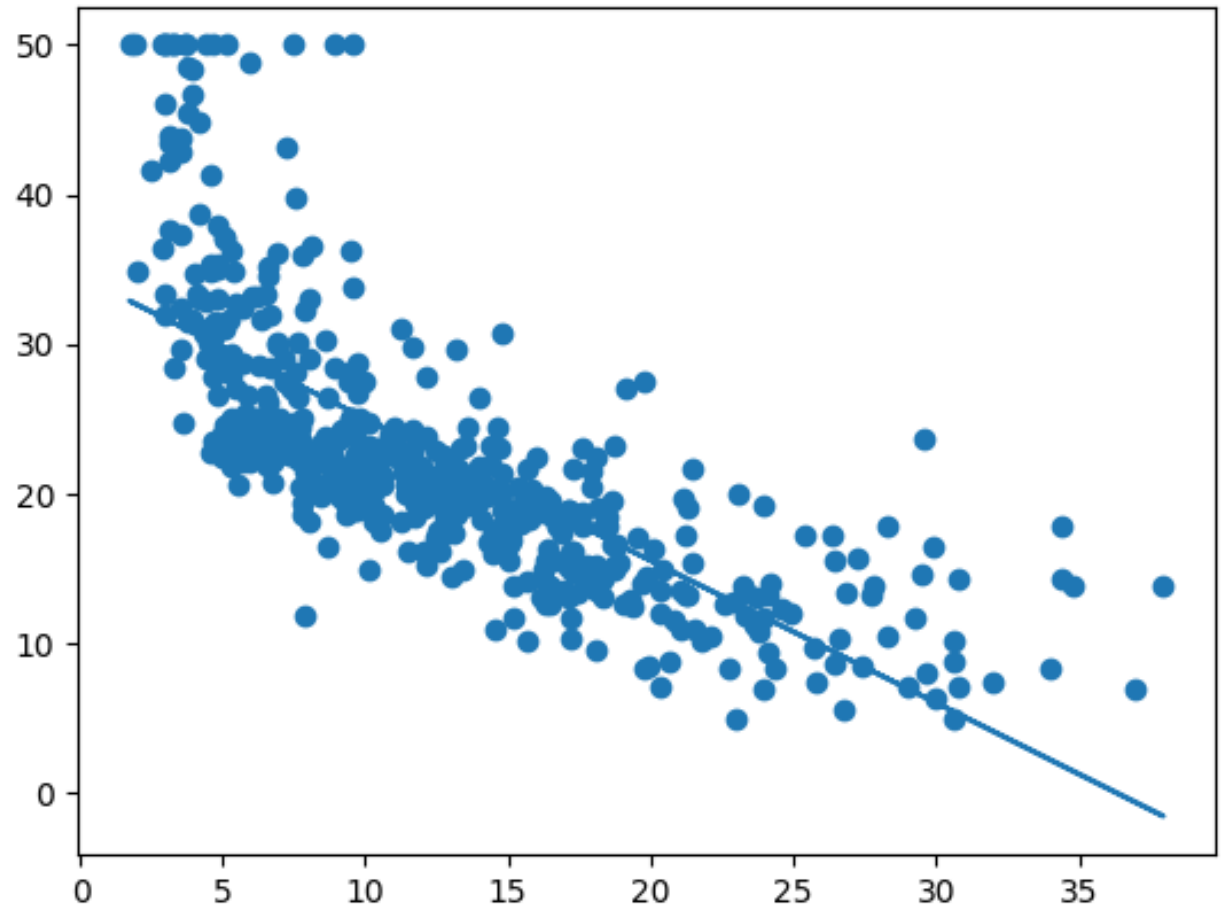
Within the scatterplot matrix, limit the view to the 3 features that have the strongest linear relationship with the target.


RM
B
LSTAT
MEDV
4
6
8
100
200
300
10
20
30
40
50
0
400
RM
B
LSTAT
MEDV

```python
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sb

os.chdir('C:/Users/Reza/Desktop/ITP449_Fall2020/Class/Files')
pd.set_option('display.max_columns', None)

boston = pd.read_csv('BostonHousing.csv')

bostonNum = boston.drop(['CHAS'], axis=1)

bostonNum['RAD'] = bostonNum['RAD'].astype(float)
bostonNum['TAX'] = bostonNum['TAX'].astype(float)

bostonNum2 = bostonNum.drop(['AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO'], axis=1)

sb.pairplot(data=bostonNum2[['RM', 'B', 'LSTAT', 'MEDV']], kind='reg')
plt.show()
```

Do the following:

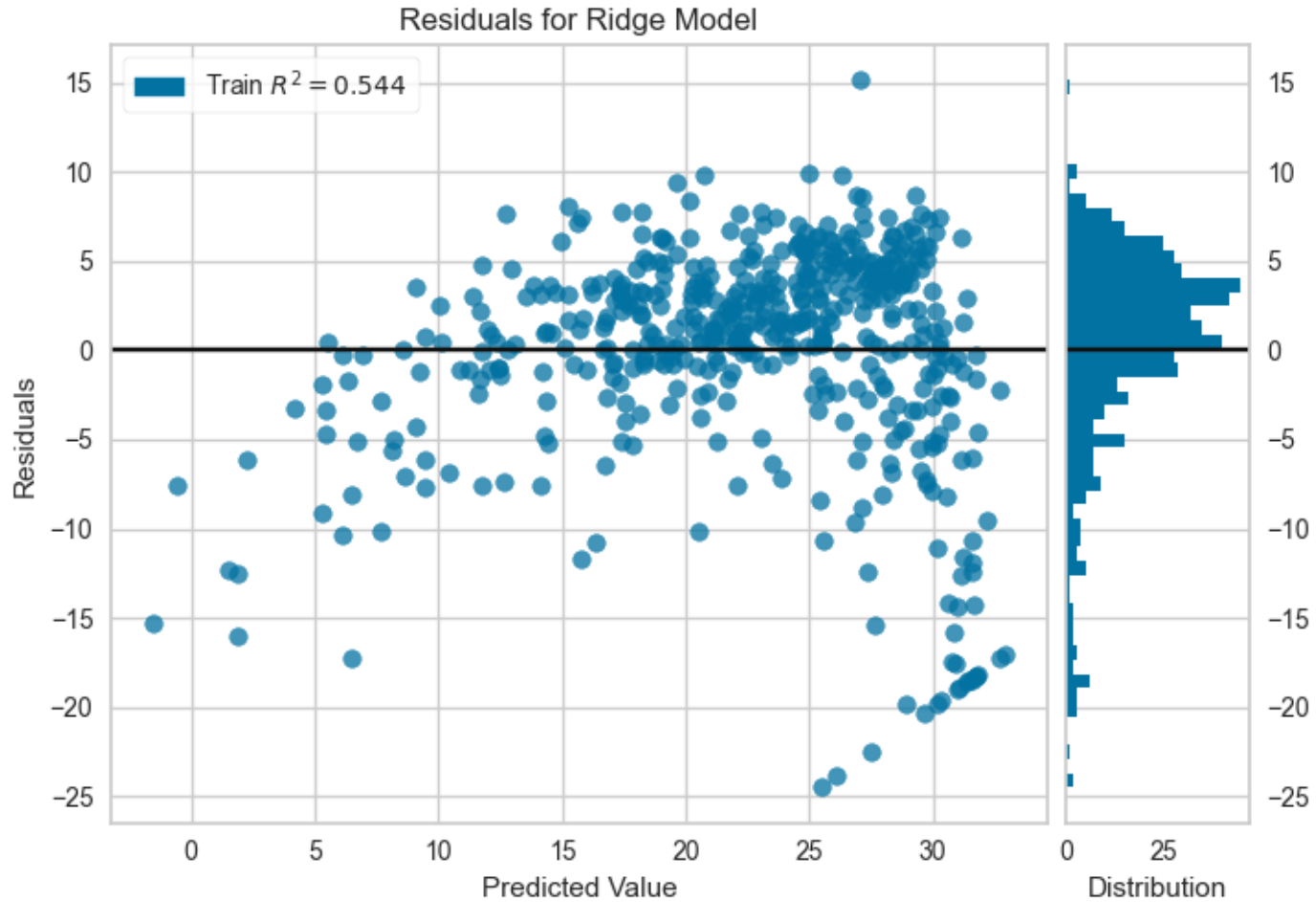
Fit a simple regression model for the feature most strongly related to the target.


50
40
30
20
10
0
0
5
10
15
20
25
30
35

```python
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

os.chdir('C:/Users/Reza/Desktop/ITP449_Fall2020/Class/Files')
pd.set_option('display.max_columns', None)

boston = pd.read_csv('BostonHousing.csv')

model = LinearRegression()
x = boston['LSTAT']
X = x[:, np.newaxis]
y = boston['MEDV']
model.fit(X, y)

y_predicted = model.predict(X)
plt.scatter(x, y)
plt.plot(x, y_predicted)
plt.show()
```

Do the following:

Plot the residuals to gain a better understanding of the fit of the model.


Residuals for Ridge Model
Train $R^2 = 0.544$
Residuals
Predicted Value
Distribution
15
10
5
0
−5
−10
−15
−20
−25
0
5
10
15
20
25
30
0
25

```python
import pandas as pd
import os
from sklearn.linear_model import Ridge
from yellowbrick.regressor import ResidualsPlot

os.chdir('C:/Users/Reza/Desktop/ITP449_Fall2020/Class/Files')
pd.set_option('display.max_columns', None)

boston = pd.read_csv('BostonHousing.csv')

x = boston['LSTAT'].values
X = x.reshape(-1, 1)
y = boston['MEDV']

# Instantiate the linear model and the visualizer
ridge = Ridge()
visualizer = ResidualsPlot(ridge)
visualizer.fit(X, y)
visualizer.show()
```

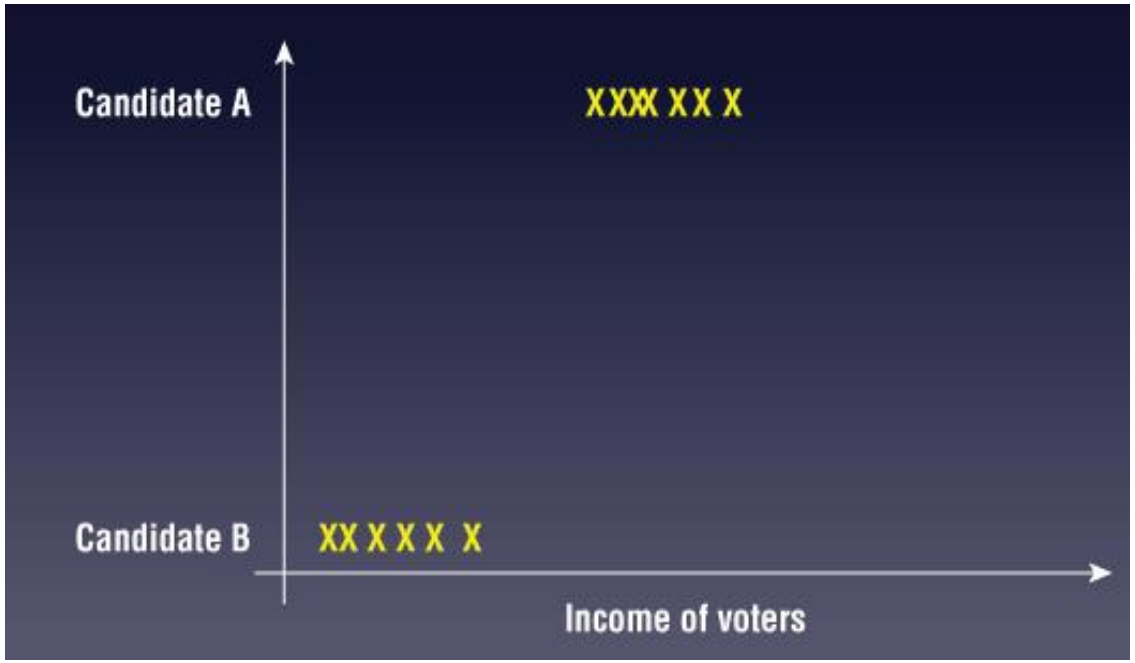
Logistic Regression

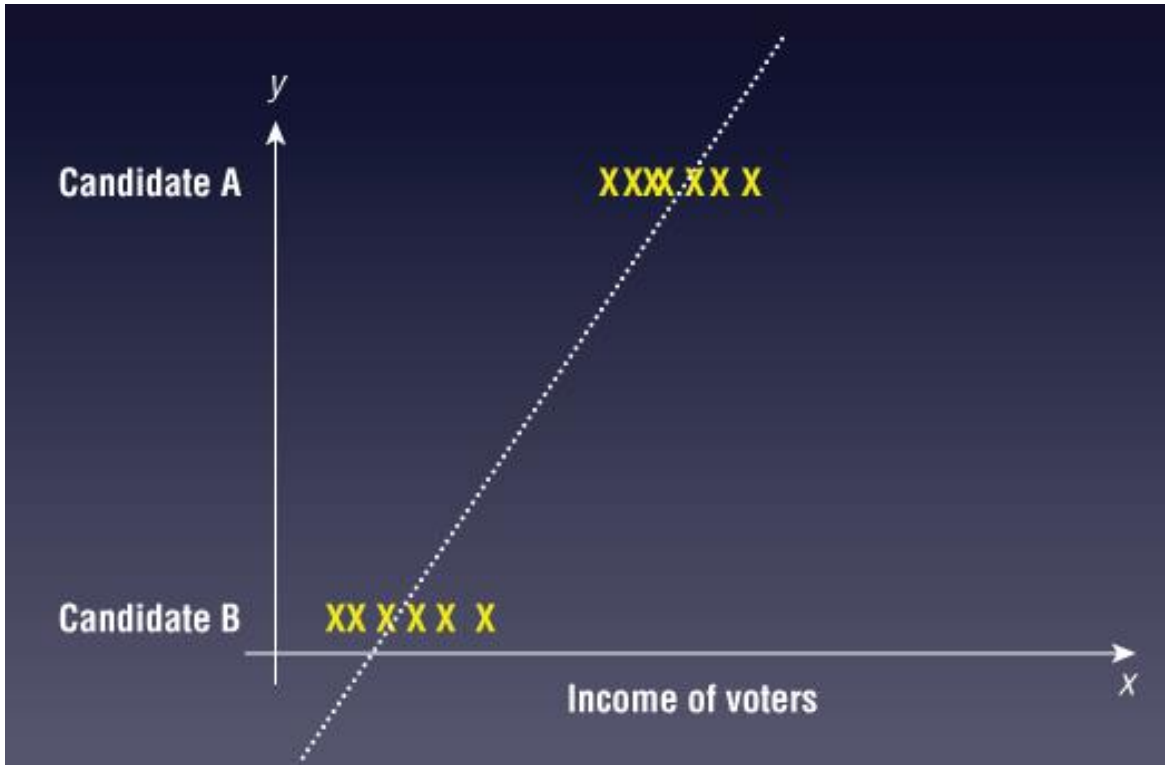
Logistic Regression

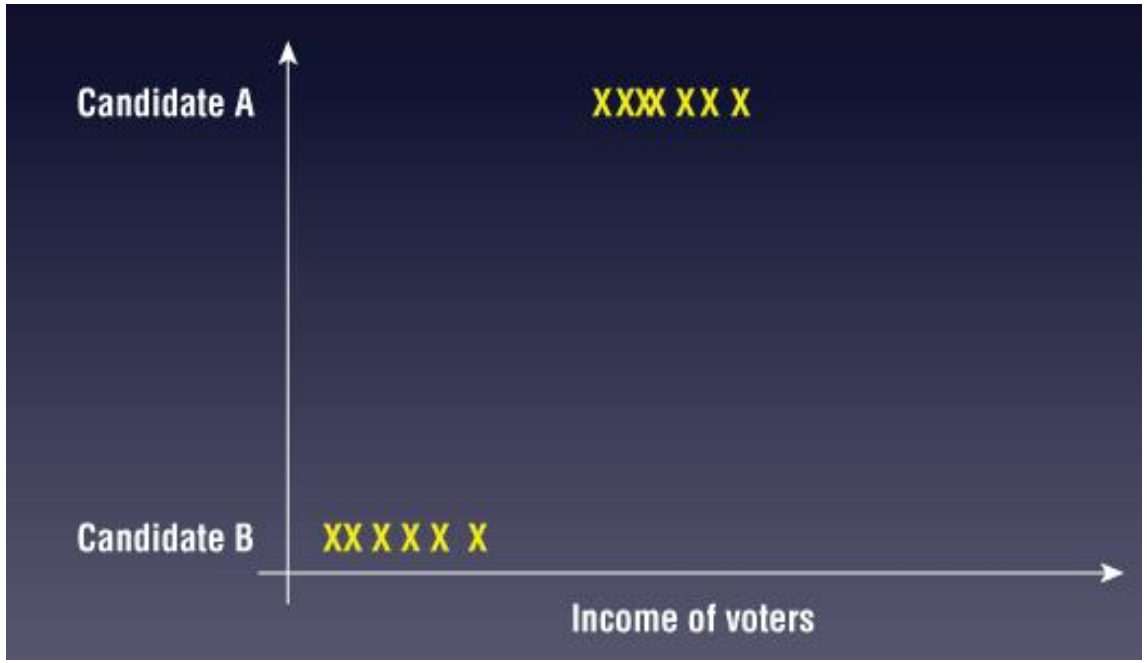
Used to predict categories (as opposed to a numerical values in Linear Regression) and solve classification machine learning problems.

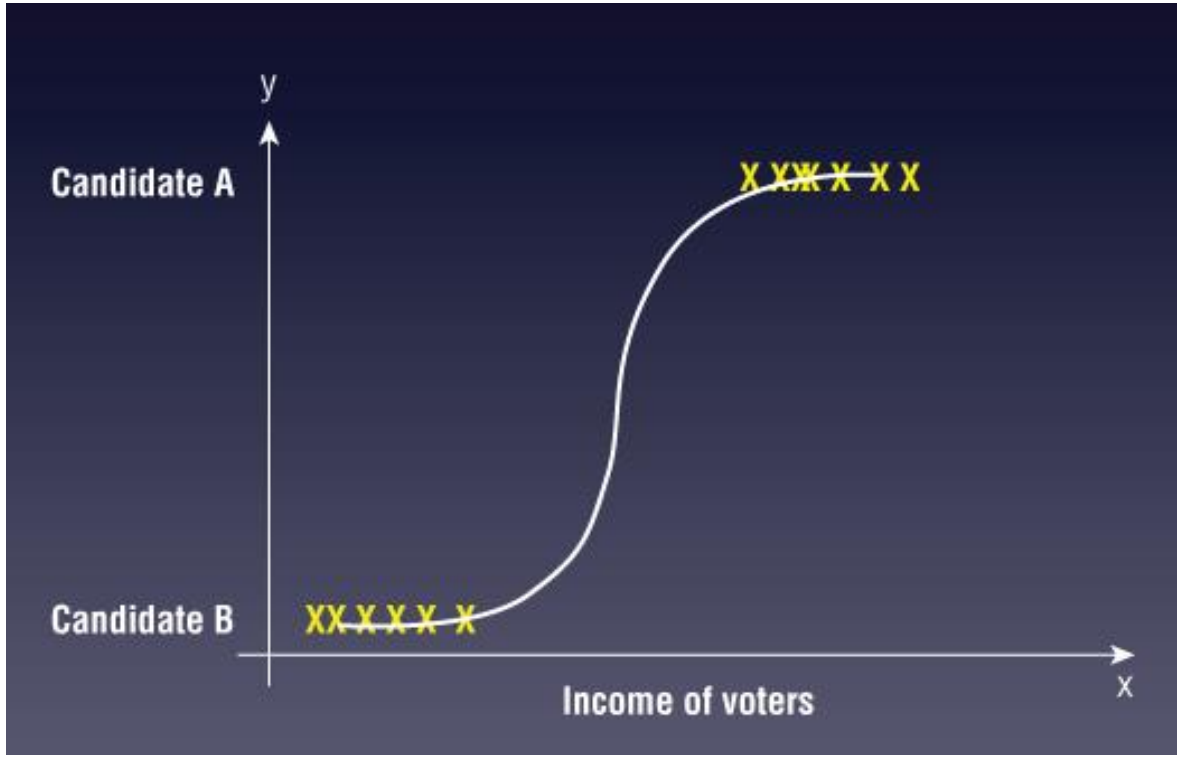
Will customer buy a product?
Will the team win the game?

	Linear regression	**Logistic regression**
Predictor variables	Continuous numeric/categorical	Continuous numeric/categorical
Output variables	Continuous numeric	Categorical
Relationship	Linear	Linear (with some transformations)


Candidate A
Candidate B
Income of voters

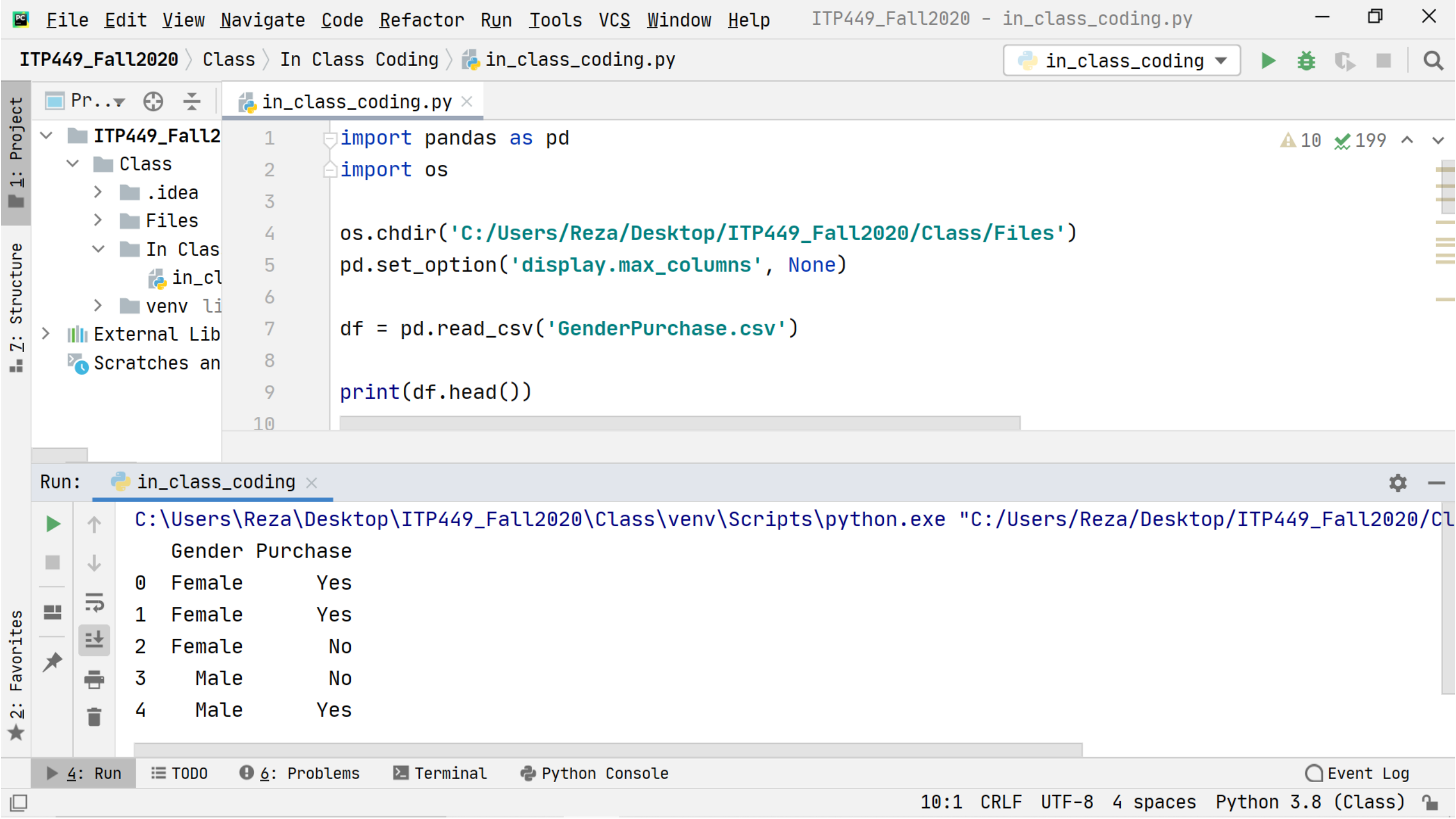

y
Candidate A
Candidate B
Income of voters
x





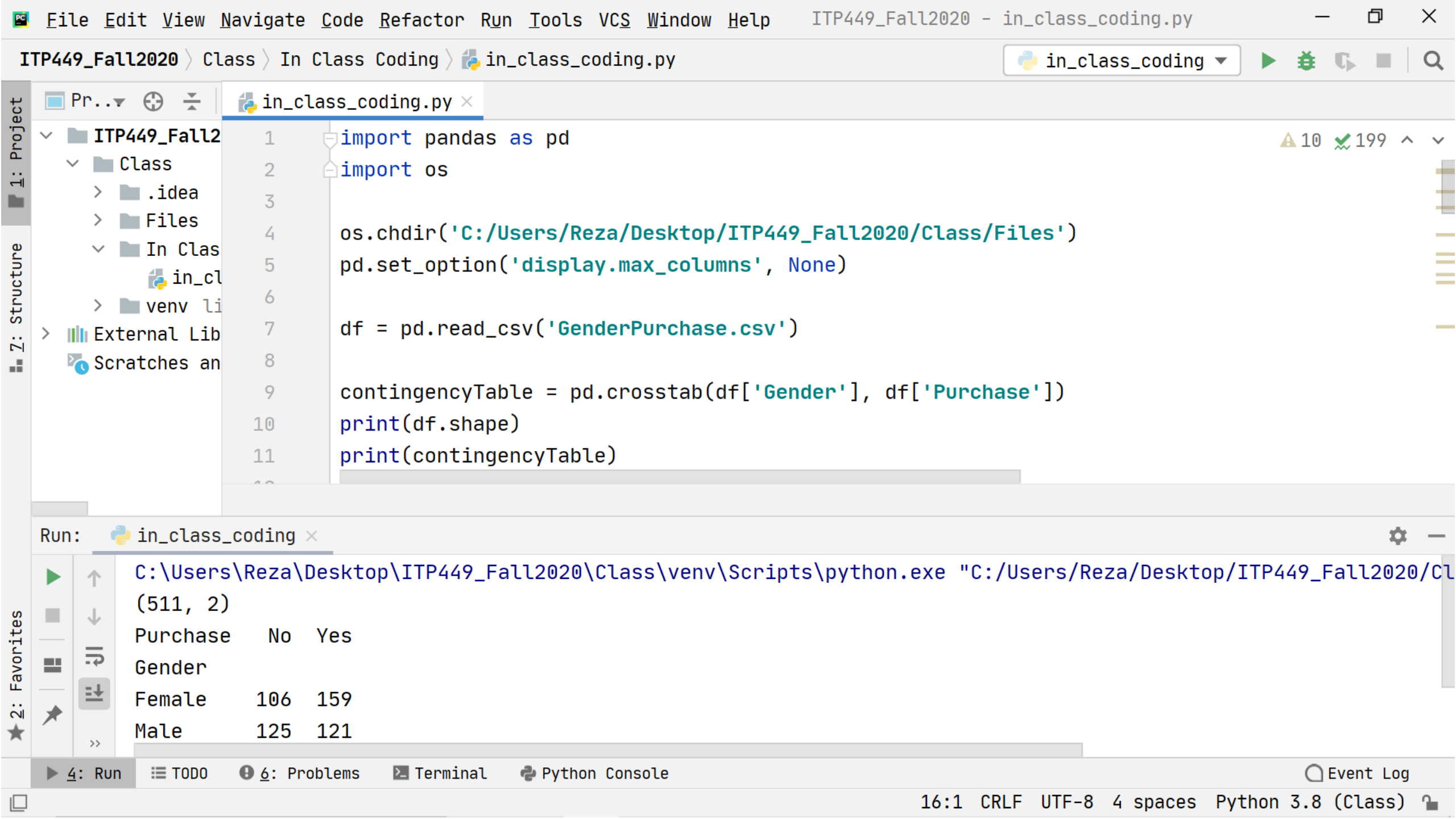
Using logistic regression, the output will be a value from 0 to 1, where anything ≤ 0.5 (known as the *threshold*) will be considered as voting for candidate B, and anything > 0.5 will be considered as voting for candidate A.

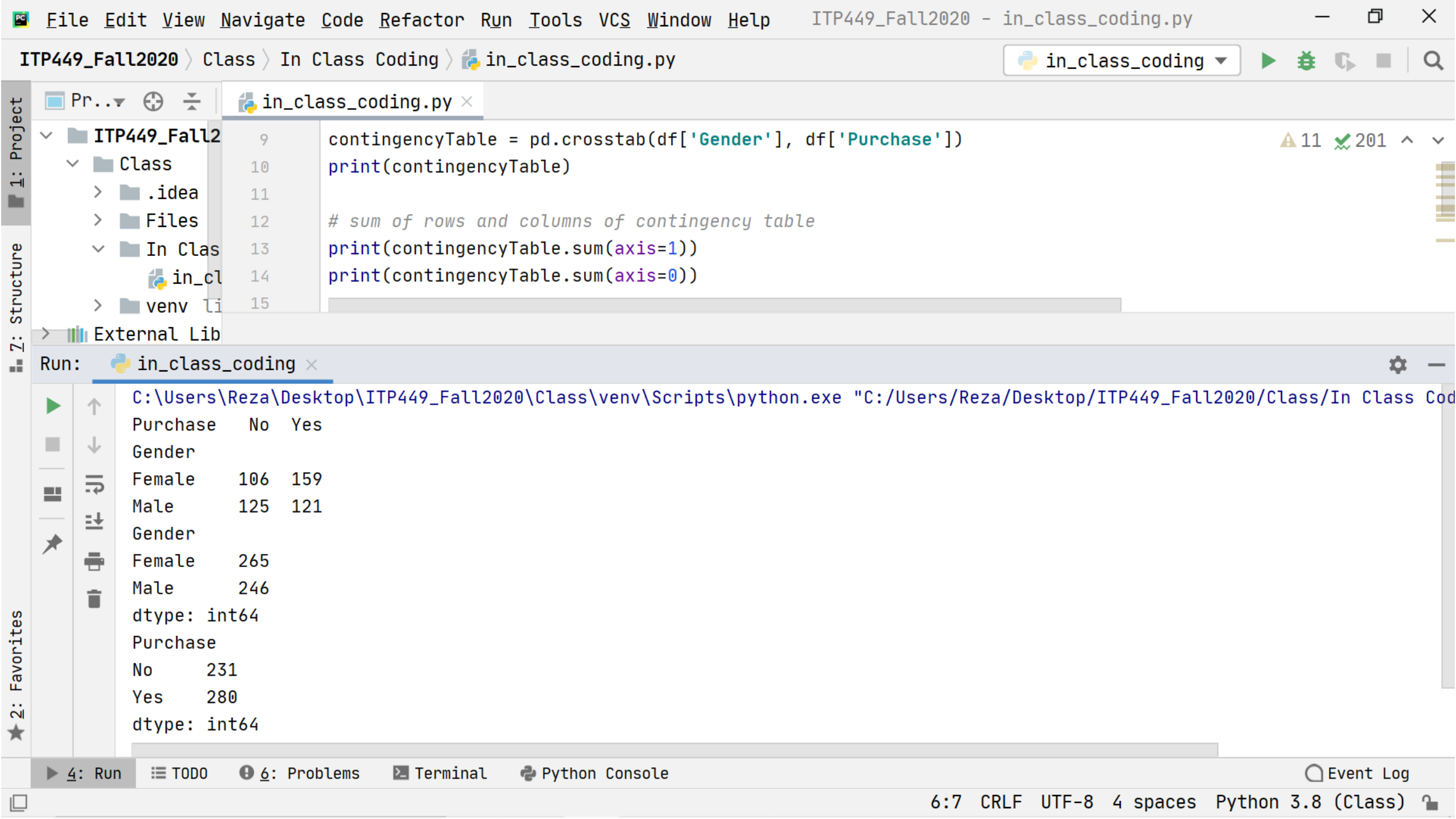
What are the chances of a customer buying a product based on their gender?


File Edit View Navigate Code Refactor Run Tools VCS Window Help
ITP449_Fall2020 - in_class_coding.py
ITP449_Fall2020 〉 Class 〉 In Class Coding 〉 in_class_coding.py
in_class_coding
in_class_coding.py
ITP449_Fall2
Class
.idea
Files
In Clas
in_cl
venv li
External Lib
Scratches an
1 import pandas as pd
2 import os
3
4 os.chdir('C:/Users/Reza/Desktop/ITP449_Fall2020/Class/Files')
5 pd.set_option('display.max_columns', None)
6
7 df = pd.read_csv('GenderPurchase.csv')
8
9 print(df.head())
10
10
199
Run: in_class_coding
C:\Users\Reza\Desktop\ITP449_Fall2020\Class\venv\Scripts\python.exe "C:/Users/Reza/Desktop/ITP449_Fall2020/Cl
Gender Purchase
0 Female Yes
1 Female Yes
2 Female No
3 Male No
4 Male Yes
1: Project
7: Structure
2: Favorites
4: Run
TODO
6: Problems
Terminal
Python Console
Event Log
10:1 CRLF UTF-8 4 spaces Python 3.8 (Class)

Contingency table

A table of the frequency of observations falling under various categories of two or more variables.


File Edit View Navigate Code Refactor Run Tools VCS Window Help
ITP449_Fall2020 - in_class_coding.py
ITP449_Fall2020 〉 Class 〉 In Class Coding 〉 in_class_coding.py
in_class_coding
in_class_coding.py
ITP449_Fall2
Class
.idea
Files
In Clas
in_cl
venv li
External Lib
Scratches an
1 import pandas as pd
2 import os
3
4 os.chdir('C:/Users/Reza/Desktop/ITP449_Fall2020/Class/Files')
5 pd.set_option('display.max_columns', None)
6
7 df = pd.read_csv('GenderPurchase.csv')
8
9 contingencyTable = pd.crosstab(df['Gender'], df['Purchase'])
10 print(df.shape)
11 print(contingencyTable)
10
199
Run: in_class_coding
C:\Users\Reza\Desktop\ITP449_Fall2020\Class\venv\Scripts\python.exe "C:/Users/Reza/Desktop/ITP449_Fall2020/Cl
(511, 2)
Purchase No Yes
Gender
Female 106 159
Male 125 121
1: Project
7: Structure
2: Favorites
4: Run
TODO
6: Problems
Terminal
Python Console
Event Log
16:1 CRLF UTF-8 4 spaces Python 3.8 (Class)


File Edit View Navigate Code Refactor Run Tools VCS Window Help
ITP449_Fall2020 - in_class_coding.py
ITP449_Fall2020 〉 Class 〉 In Class Coding 〉 in_class_coding.py
in_class_coding
in_class_coding.py
ITP449_Fall2
Class
.idea
Files
In Clas
in_cl
venv
External Lib
9 contingencyTable = pd.crosstab(df['Gender'], df['Purchase'])
10 print(contingencyTable)
11
12 # sum of rows and columns of contingency table
13 print(contingencyTable.sum(axis=1))
14 print(contingencyTable.sum(axis=0))
15
11
201
Run: in_class_coding
C:\Users\Reza\Desktop\ITP449_Fall2020\Class\venv\Scripts\python.exe "C:/Users/Reza/Desktop/ITP449_Fall2020/Class/In Class Cod
Purchase No Yes
Gender
Female 106 159
Male 125 121
Gender
Female 265
Male 246
dtype: int64
Purchase
No 231
Yes 280
dtype: int64
4: Run
TODO
6: Problems
Terminal
Python Console
Event Log
6:7 CRLF UTF-8 4 spaces Python 3.8 (Class)

File Edit View Navigate Code Refactor Run Tools VCS Window Help ITP449_Fall2020 - in_class_coding.py

ITP449_Fall2020 〉 Class 〉 In Class Coding 〉 in_class_coding.py

in_class_coding

1: Project | 7: Structure | 2: Favorites

Pr...

ITP449_Fall2
Class
.idea
Files
In Clas
in_cl
venv li
External Lib

in_class_coding.py

```
9   contingencyTable = pd.crosstab(df['Gender'], df['Purchase'])
10  print(contingencyTable)
11
12  # contingency table as percentages of gender total
13  print(contingencyTable.astype('float').div(contingencyTable.sum(axis=1), axis=0))
14
```

12 201

div: Floating division of the dataframe and *other* element-wise
DataFrame.div(other, axis='columns')

Run: in_class_coding

```
C:\Users\Reza\Desktop\ITP449_Fall2020\Class\venv\Scripts\python.exe "C:/Users/Reza/Desktop/ITP449_Fall2020/Cl
Purchase    No   Yes
Gender
Female     106   159
Male       125   121
Purchase         No       Yes
Gender
Female     0.40000  0.60000
Male       0.50813  0.49187

Process finished with exit code 0
```

4: Run | TODO | 6: Problems | Terminal | Python Console | Event Log

12:1 CRLF UTF-8 4 spaces Python 3.8 (Class)

Conditional Probability

The probability of a certain event happening, given that a certain related event is true or has already happened.

$$Probability(Purchase \mid Male) = \frac{Total\ number\ of\ \ purchases\ by\ males}{Total\ number\ of\ \ males\ in\ the\ group}$$

$$Probability\,(Purchase \mid Male) = 121 / 246 = 0.49$$
$$Probability\,(Not\ Purchase \mid Male) = 125 / 246 = 1 - 0.49 = 0.51$$
$$Probability\,(Purchase \mid Female) = 159 / 265 = 0.60$$
$$Probability\,(Not\ Purchase \mid Female) = 106 / 265 = 1 - 0.60 = 0.40$$

Odds Ratio

A ratio of odds of success (*purchase*) for each group (*male and female*).

Odds of success: The ratio of probability of successes (*purchases*) to the probability of failures (*non-purchases*).

$$Odds\ of\ purchase\ by\ males = P_m / (1 - P_m)$$
$$Odds\ of\ purchase\ by\ females = P_f / (1 - P_f)$$

$$Odds\ ratio(for\ males) = Odds\ of\ success\ for\ males / Odds\ of\ success\ for\ females$$
$$= (121/125) / (159/106) = 0.64$$
$$Odds\ ratio(for\ females) = Odds\ of\ Purchase\ for\ female / Odds\ of\ purchase\ by\ males$$
$$= (159/106) / (121/125) = 1.54$$

Odds Ratio Interpretation

- If the odds of success for a group is more than 1, then it is more likely for that group to be successful. The higher the odds, the better the chances of success.
- If the odds of success is less than 1, then it is more likely to get a failure. The lower the odds, the higher the chances of failure.
- The odds can range from 0 to infinity.
- If the odds ratio = 1, then no association between the two variables.
- If odds ratio > 1, then success of group is more likely.
- If odds ratio < 1, then success of group is less likely
- The odds ratio for one group is the reciprocal of the odds ratio of the other group.

More probable for female to purchase than male.

from *Linear* to *Logistic* Regression

$$Y = a + b * X$$

Y range: *–inf to +inf* ***X*** range: *–inf to +inf*

$$P = a + b * X$$

P range: *0 to 1* ***X*** range: *–inf to +inf*

P conditional probability of success given ***X***

$$P/1-P = a + b * X$$

P/1-P range: *0 to +inf* ***X*** range: *–inf to +inf*

P/1-P odds of success given ***X***

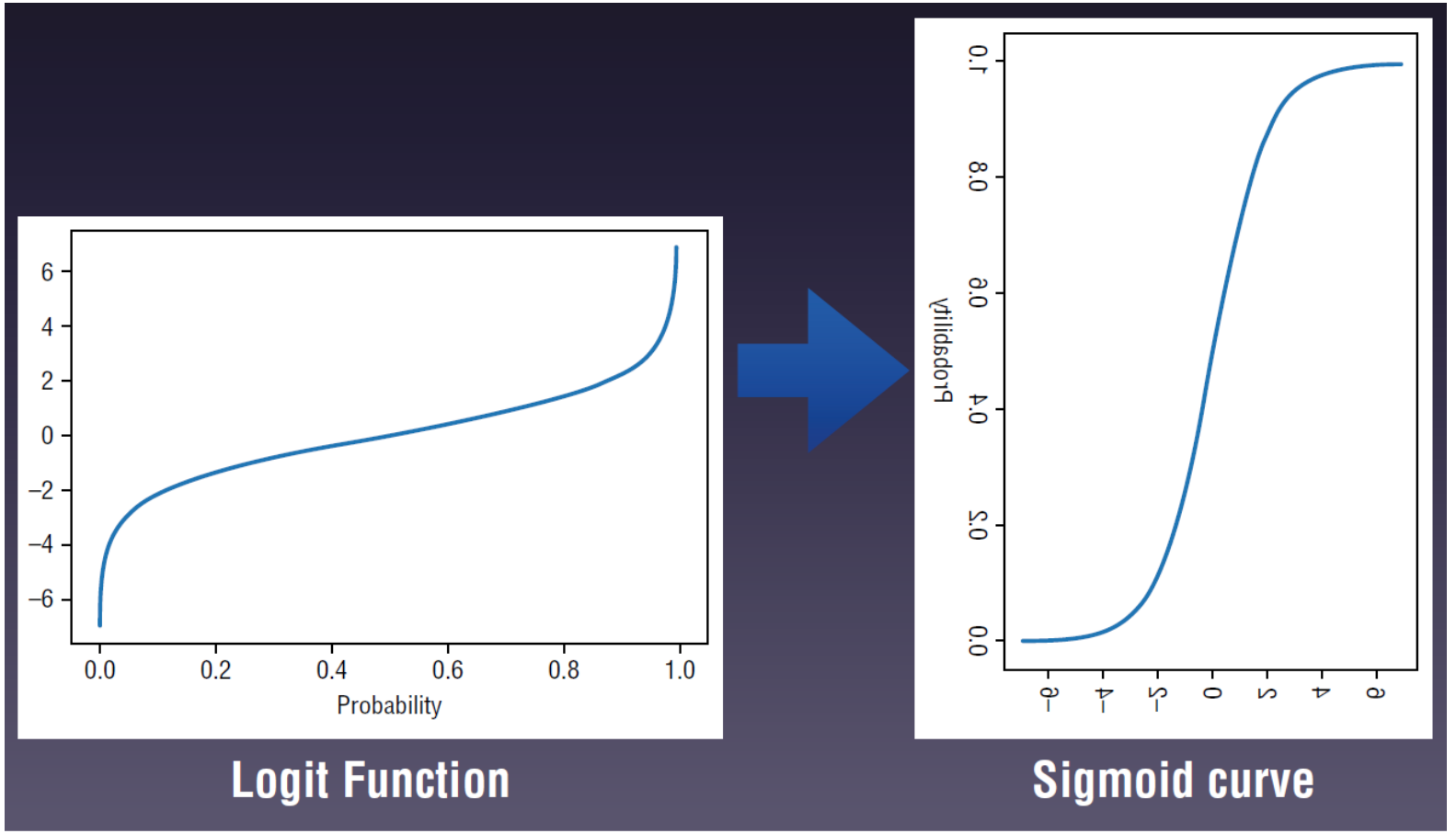
$$\log(P/1-P) = a + b * X$$

log(P/1-P) range: *-inf to +inf* ***X*** range: *–inf to +inf*

$$P = \frac{e^{a+b*X}}{1+e^{a+b*X}} = \frac{1}{1+e^{-(a+b*X)}}$$

$$P = \frac{1}{1+e^{-(a+b_1*X_1+b_2*X_2+b_3*X_3+\ldots+bn*Xn)}}$$

- If *a+b*X* is very small, then *P* approaches 0
- If *a+b*X* is very large, then *P* approaches 1
- If *a+b*X* is 0 then *P=0.5*



$$L = \ln\left(\frac{P}{1-P}\right)$$

$$P = \frac{1}{(1 + e^{-(L)})}$$

Diabetes Example:

https://www.kaggle.com/uciml/pima-indians-diabetes-database

Context:

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

```python
import pandas as pd
import os

os.chdir('C:/Users/Reza/Desktop/ITP449_Fall2020/Class/Files')
pd.set_option('display.max_columns', None)

col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']
pima = pd.read_csv('diabetes.csv', header=1, names=col_names)
print(pima.head())
```

16 203

Run: in_class_coding

```
C:\Users\Reza\Desktop\ITP449_Fall2020\Class\venv\Scripts\python.exe "C:/Users/Reza/Desktop/ITP449_Fall2020/Cl
   pregnant  glucose  bp  skin  insulin   bmi  pedigree  age  label
0         1       85  66    29        0  26.6     0.351   31      0
1         8      183  64     0        0  23.3     0.672   32      1
2         1       89  66    23       94  28.1     0.167   21      0
3         0      137  40    35      168  43.1     2.288   33      1
4         5      116  74     0        0  25.6     0.201   30      0
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help ITP449_Fall2020 - in_class_coding.py

ITP449_Fall2020 〉 Class 〉 In Class Coding 〉 in_class_coding.py

in_class_coding

1: Project

in_class_coding.py

```
8   pima = pd.read_csv('diabetes.csv', header=1, names=col_names)
9   print(pima.head())
10
11  feature_cols = ['pregnant', 'insulin', 'bmi', 'age', 'glucose', 'bp', 'pedigree']
12  X = pima[feature_cols] # features
13  y = pima.label # target variable
14
15  print(X.head())
16  print(y.head())
```

17 206

Run: in_class_coding

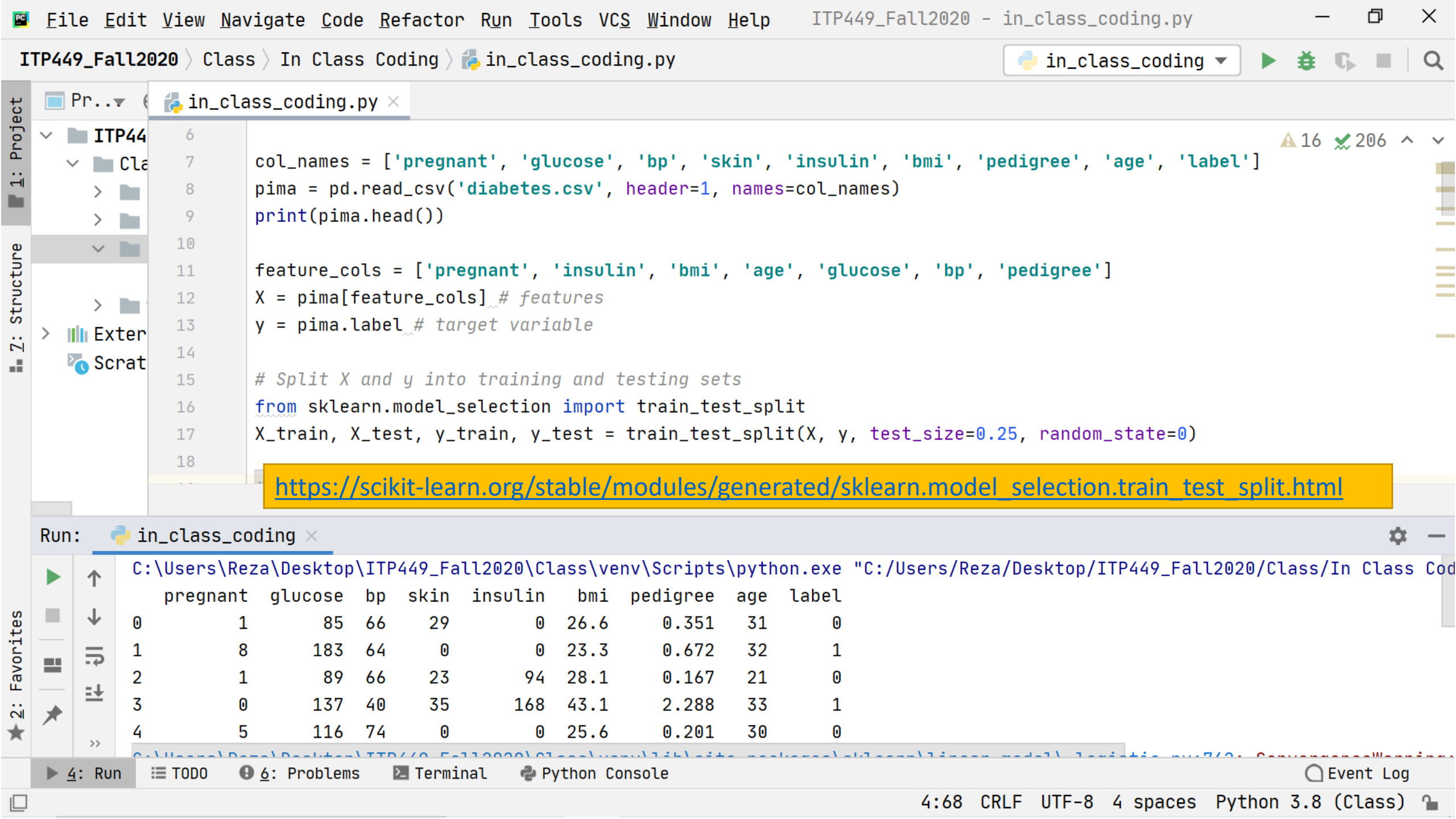
```
   pregnant  insulin   bmi  age  glucose  bp  pedigree
0         1        0  26.6   31       85  66     0.351
1         8        0  23.3   32      183  64     0.672
2         1       94  28.1   21       89  66     0.167
3         0      168  43.1   33      137  40     2.288
4         5        0  25.6   30      116  74     0.201
0    0
1    1
2    0
3    1
4    0
```

7: Structure

2: Favorites

4: Run TODO 6: Problems Terminal Python Console Event Log

18:1 CRLF UTF-8 4 spaces Python 3.8 (Class)


File Edit View Navigate Code Refactor Run Tools VCS Window Help
ITP449_Fall2020 - in_class_coding.py
ITP449_Fall2020 〉 Class 〉 In Class Coding 〉 in_class_coding.py
in_class_coding
in_class_coding.py
16 206
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']
pima = pd.read_csv('diabetes.csv', header=1, names=col_names)
print(pima.head())
feature_cols = ['pregnant', 'insulin', 'bmi', 'age', 'glucose', 'bp', 'pedigree']
X = pima[feature_cols] # features
y = pima.label # target variable
# Split X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
Run: in_class_coding
C:\Users\Reza\Desktop\ITP449_Fall2020\Class\venv\Scripts\python.exe "C:/Users/Reza/Desktop/ITP449_Fall2020/Class/In Class Cod
pregnant glucose bp skin insulin bmi pedigree age label
0 1 85 66 29 0 26.6 0.351 31 0
1 8 183 64 0 0 23.3 0.672 32 1
2 1 89 66 23 94 28.1 0.167 21 0
3 0 137 40 35 168 43.1 2.288 33 1
4 5 116 74 0 0 25.6 0.201 30 0
4: Run
TODO
6: Problems
Terminal
Python Console
Event Log
4:68 CRLF UTF-8 4 spaces Python 3.8 (Class)

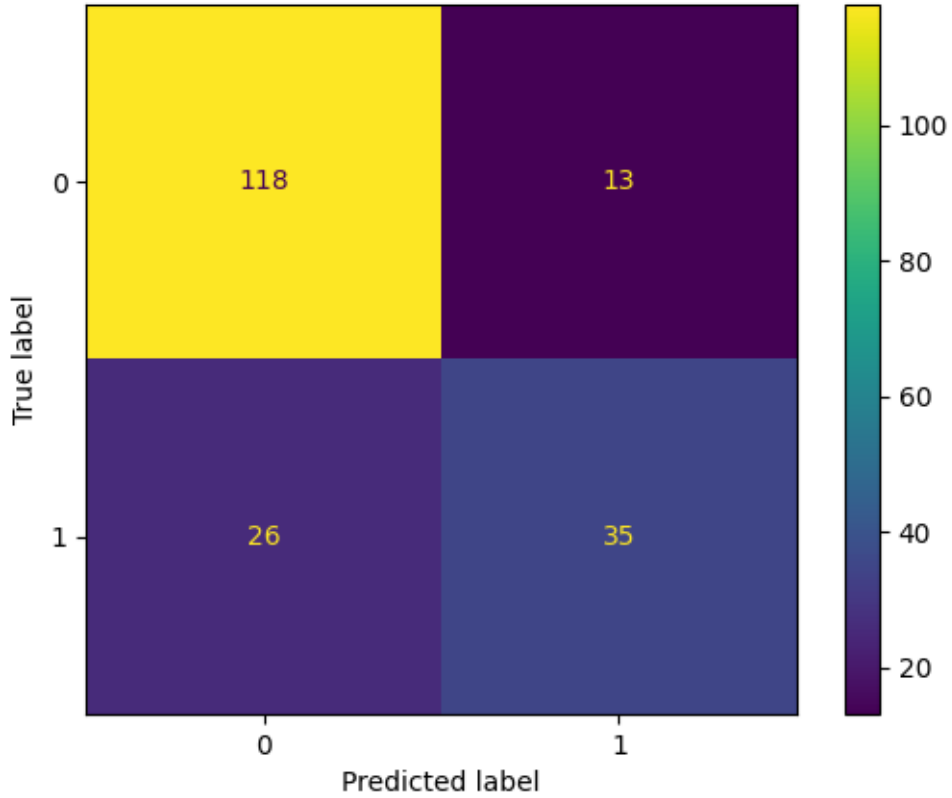
in_class_coding.py

```
19  # import the class
20  from sklearn.linear_model import LogisticRegression
21
22  # instantiate the model (using default parameters)
23  logReg = LogisticRegression()
24
25  # fit the model with data
26  logReg.fit(X_train, y_train)
27
28  # make predictions
29  y_pred = logReg.predict(X_test)
30
31  # evaluate the performance
32  from sklearn import metrics
33  cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
34
35  print(cnf_matrix)
36  print('Accuracy:', metrics.accuracy_score(y_test, y_pred))
37
```

16 206

Run: in_class_coding

```
[[118  13]
 [ 26  35]]
Accuracy: 0.796875
```



$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FN} + \mathrm{FP}} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{P} + \mathrm{N}}$$

Confusion Matrix	Predicted Response	
Actual Response	Yes	No
Yes	True positives (hit)	False negatives (miss)
No	False positives (false alarm)	True negatives (correct rejection)

Bank Marketing Data Set

Download: Data Folder, Data Set Description

Abstract: The data is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit (variable y).

Data Set Characteristics:	Multivariate	**Number of Instances:**	45211	**Area:**	Business
Attribute Characteristics:	Real	**Number of Attributes:**	17	**Date Donated**	2012-02-14
Associated Tasks:	Classification	**Missing Values?**	N/A	**Number of Web Hits:**	1297580

Source:

[Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014

Data Set Information:

The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

There are four datasets:
1) bank-additional-full.csv with all examples (41188) and 20 inputs, ordered by date (from May 2008 to November 2010), very close to the data analyzed in [Moro et al., 2014]
2) bank-additional.csv with 10% of the examples (4119), randomly selected from 1), and 20 inputs.
3) bank-full.csv with all examples and 17 inputs, ordered by date (older version of this dataset with less inputs).
4) bank.csv with 10% of the examples and 17 inputs, randomly selected from 3 (older version of this dataset with less inputs).
The smallest datasets are provided to test more computationally demanding machine learning algorithms (e.g., SVM).

The classification goal is to predict if the client will subscribe (yes/no) a term deposit (variable y).

Attribute Information:

Input variables:
bank client data:
1 - age (numeric)
2 - job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
4 - education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
5 - default: has credit in default? (categorical: 'no','yes','unknown')
6 - housing: has housing loan? (categorical: 'no','yes','unknown')
7 - loan: has personal loan? (categorical: 'no','yes','unknown')
related with the last contact of the current campaign:
8 - contact: contact communication type (categorical: 'cellular','telephone')
9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
10 - day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
11 - duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.
other attributes:
12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
14 - previous: number of contacts performed before this campaign and for this client (numeric)
15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')
social and economic context attributes
16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)
17 - cons.price.idx: consumer price index - monthly indicator (numeric)
18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)
19 - euribor3m: euribor 3 month rate - daily indicator (numeric)
20 - nr.employed: number of employees - quarterly indicator (numeric)

Output variable (desired target):
21 - y - has the client subscribed a term deposit? (binary: 'yes','no')

Do the following:

Create a DataFrame variable containing the banking data CSV file.

https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

```
   age          job  marital           education  default housing loan  \
0   44  blue-collar  married            basic.4y  unknown     yes   no
1   53   technician  married             unknown       no      no   no
2   28   management   single  university.degree       no     yes   no
3   39     services  married         high.school       no      no   no
4   55      retired  married            basic.4y       no     yes   no

    contact month day_of_week  duration  campaign  pdays  previous  \
0  cellular   aug         thu       210         1    999         0
1  cellular   nov         fri       138         1    999         0
2  cellular   jun         thu       339         3      6         2
3  cellular   apr         fri       185         2    999         0
4  cellular   aug         fri       137         1      3         1

      poutcome  emp_var_rate  cons_price_idx  cons_conf_idx  euribor3m  \
0  nonexistent           1.4          93.444          -36.1      4.963
1  nonexistent          -0.1          93.200          -42.0      4.021
2      success          -1.7          94.055          -39.8      0.729
3  nonexistent          -1.8          93.075          -47.1      1.405
4      success          -2.9          92.201          -31.4      0.869

   nr_employed  y
0       5228.1  0
1       5195.8  0
2       4991.6  1
3       5099.1  0
4       5076.2  1
(41188, 21)
['age' 'job' 'marital' 'education' 'default' 'housing' 'loan' 'contact'
 'month' 'day_of_week' 'duration' 'campaign' 'pdays' 'previous' 'poutcome'
 'emp_var_rate' 'cons_price_idx' 'cons_conf_idx' 'euribor3m' 'nr_employed'
 'y']
```

```python
import pandas as pd
import os

os.chdir('C:/Users/Reza/Desktop/ITP449_Fall2020/Class/Files')
pd.set_option('display.max_columns', None)

bankData = pd.read_csv('banking.csv', header=0)

print(bankData.head())
print(bankData.shape)
print(bankData.columns.values)
```

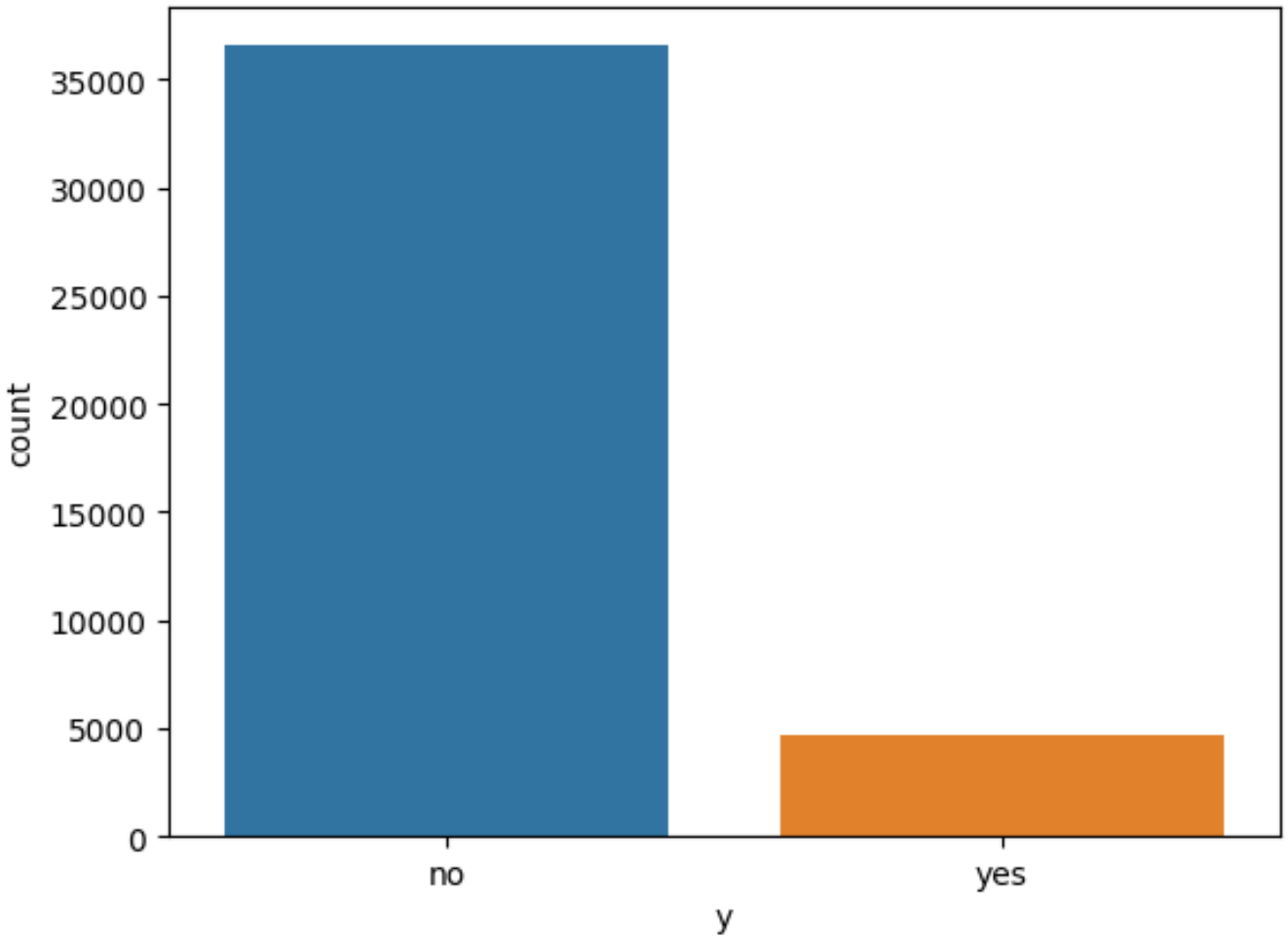

Run: in_class_coding

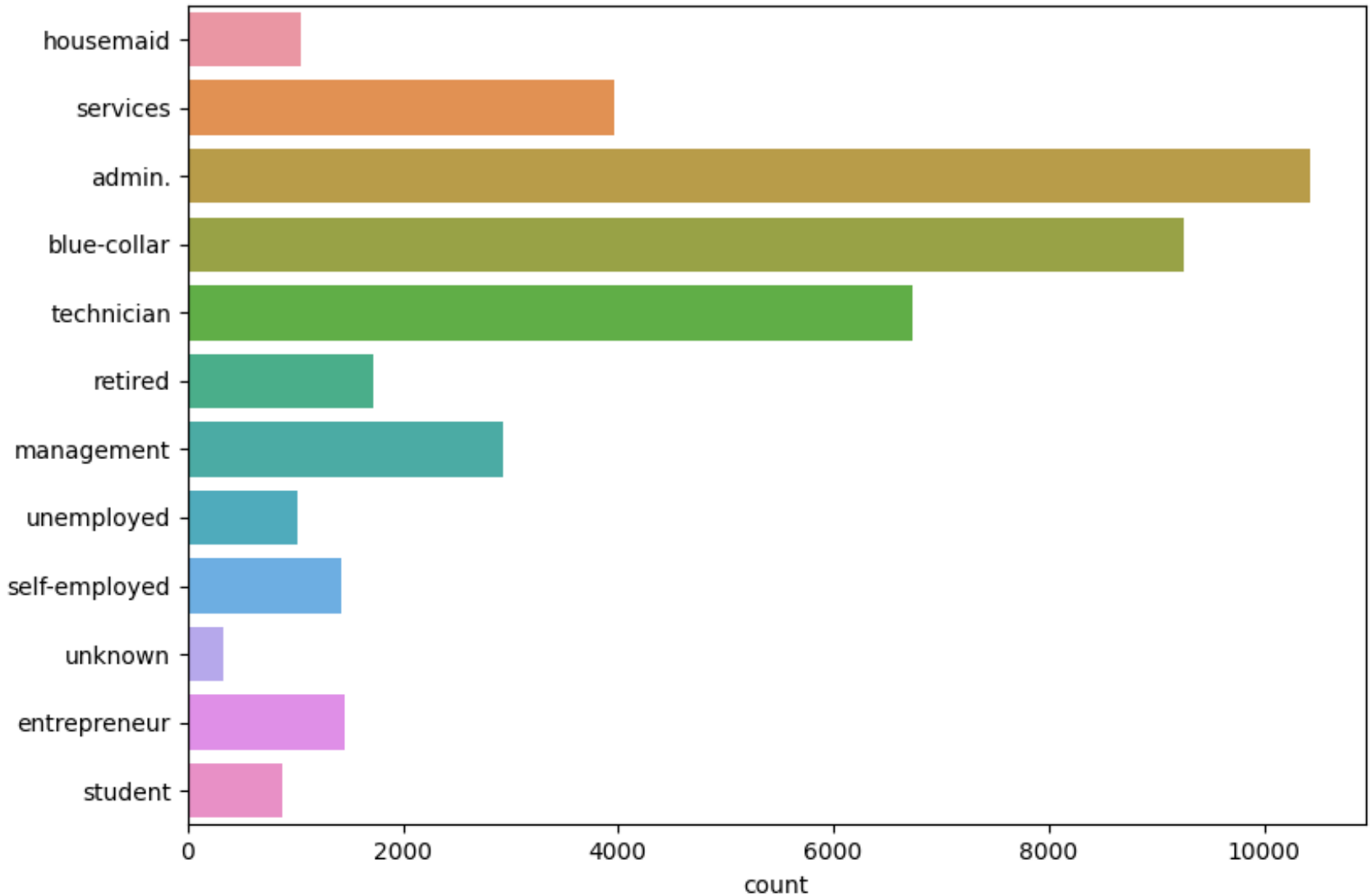
```
(41188, 21)
['age' 'job' 'marital' 'education' 'default' 'housing' 'loan' 'contact'
 'month' 'day_of_week' 'duration' 'campaign' 'pdays' 'previous' 'poutcome'
 'emp.var.rate' 'cons.price.idx' 'cons.conf.idx' 'euribor3m' 'nr.employed'
 'y']
```

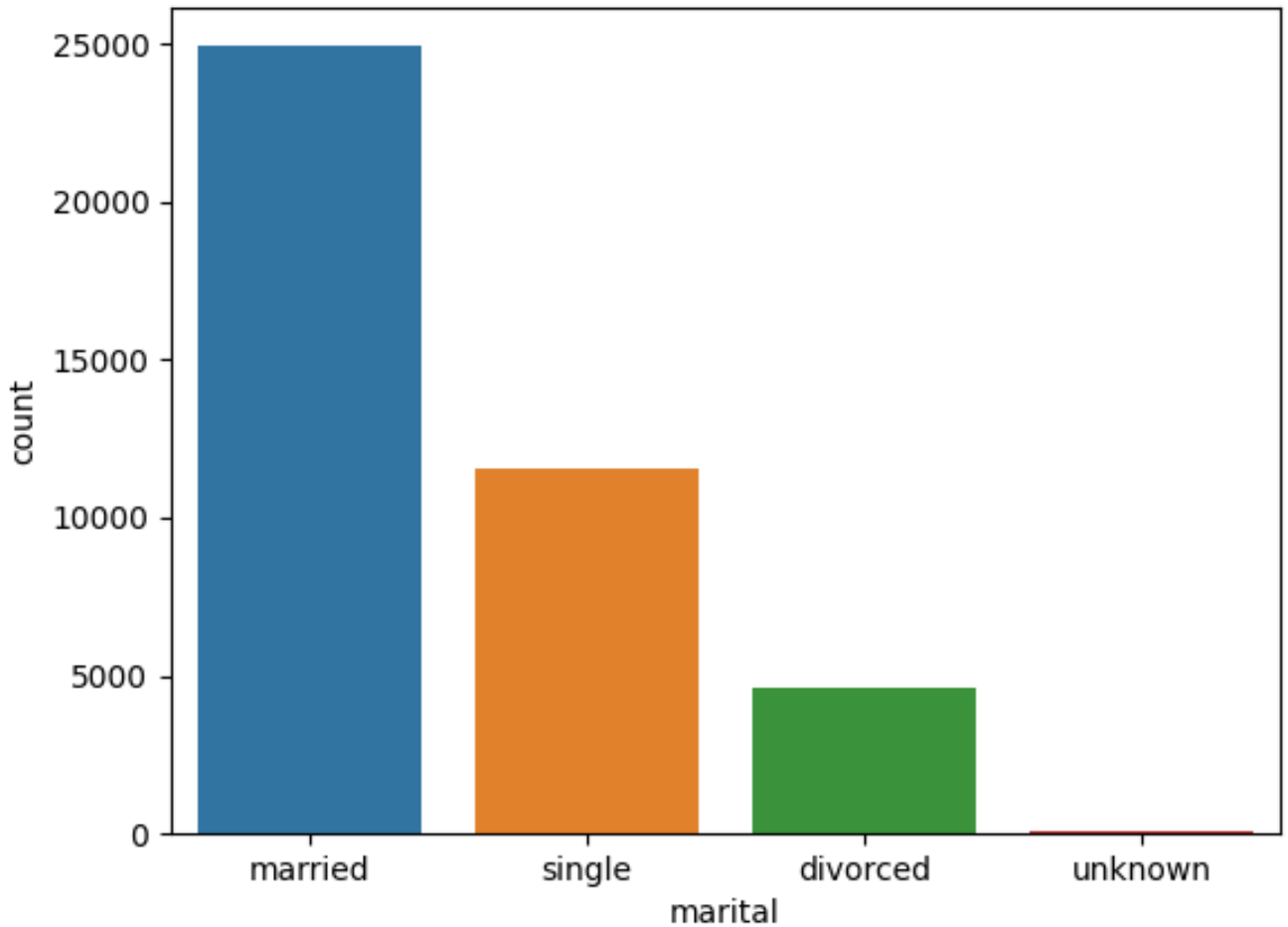
Do the following:

Explore the dataset and determine what the dependent variable (target array) is. Analyze the dependent variable as well as some key independent variables and make sure there are no missing values.

```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp_var_rate      0
cons_price_idx    0
cons_conf_idx     0
euribor3m         0
nr_employed       0
y                 0
dtype: int64
```


35000
30000
25000
20000
15000
10000
5000
0
count
no
yes
y


housemaid
services
admin.
blue-collar
technician
retired
management
unemployed
self-employed
unknown
entrepreneur
student
0
2000
4000
6000
8000
10000
count


25000
20000
15000
10000
5000
0
count
married
single
divorced
unknown
marital

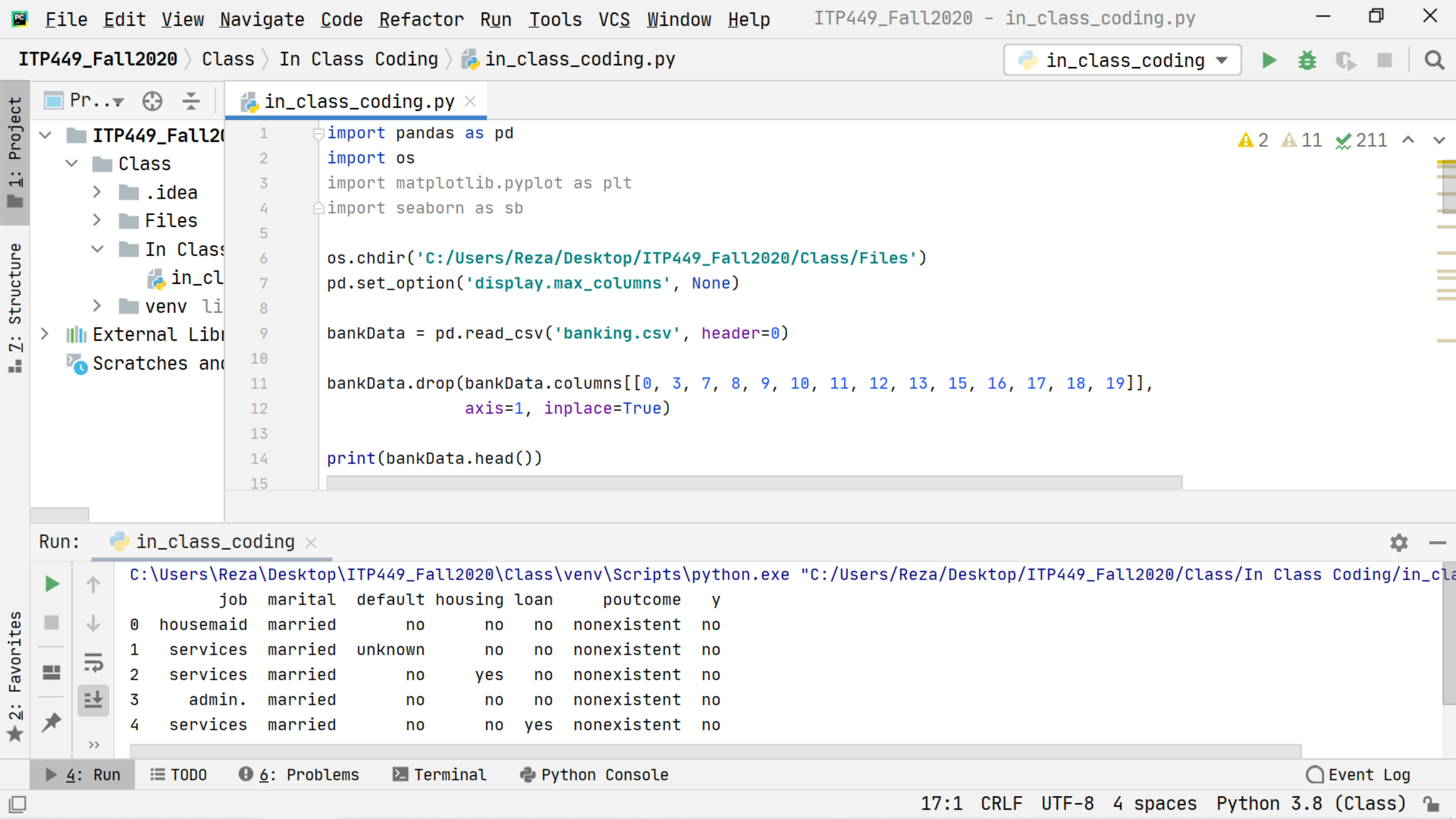
```python
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sb

os.chdir('C:/Users/Reza/Desktop/ITP449_Fall2020/Class/Files')
pd.set_option('display.max_columns', None)

bankData = pd.read_csv('banking.csv', header=0)

print(bankData.isnull().sum())

plt.figure(1)
sb.countplot(x='y', data=bankData)

plt.figure(2)
sb.countplot(y='job', data=bankData)

plt.figure(3)
sb.countplot(x='marital', data=bankData)

plt.show()
```

Do the following:

Determine what features to include in the model. Remove the features you won’t include from the dataFrame.

```
   age          job  marital           education  default housing loan  \
0   44  blue-collar  married            basic.4y  unknown     yes   no
1   53   technician  married             unknown       no      no   no
2   28   management   single  university.degree       no     yes   no
3   39     services  married         high.school       no      no   no
4   55      retired  married            basic.4y       no     yes   no

    contact month day_of_week  duration  campaign  pdays  previous  \
0  cellular   aug         thu       210         1    999         0
1  cellular   nov         fri       138         1    999         0
2  cellular   jun         thu       339         3      6         2
3  cellular   apr         fri       185         2    999         0
4  cellular   aug         fri       137         1      3         1

      poutcome  emp_var_rate  cons_price_idx  cons_conf_idx  euribor3m  \
0  nonexistent           1.4          93.444          -36.1      4.963
1  nonexistent          -0.1          93.200          -42.0      4.021
2      success          -1.7          94.055          -39.8      0.729
3  nonexistent          -1.8          93.075          -47.1      1.405
4      success          -2.9          92.201          -31.4      0.869

   nr_employed  y
0       5228.1  0
1       5195.8  0
2       4991.6  1
3       5099.1  0
4       5076.2  1
(41188, 21)
['age' 'job' 'marital' 'education' 'default' 'housing' 'loan' 'contact'
 'month' 'day_of_week' 'duration' 'campaign' 'pdays' 'previous' 'poutcome'
 'emp_var_rate' 'cons_price_idx' 'cons_conf_idx' 'euribor3m' 'nr_employed'
 'y']
```

```
   age          job  marital            education  default housing loan  \
0   44  blue-collar  married             basic.4y  unknown     yes   no
1   53   technician  married              unknown       no      no   no
2   28   management   single  university.degree       no     yes   no
3   39     services  married          high.school       no      no   no
4   55      retired  married             basic.4y       no     yes   no

    contact month day_of_week  duration  campaign  pdays  previous  \
0  cellular   aug         thu       210         1    999         0
1  cellular   nov         fri       138         1    999         0
2  cellular   jun         thu       339         3      6         2
3  cellular   apr         fri       185         2    999         0
4  cellular   aug         fri       137         1      3         1

      poutcome  emp_var_rate  cons_price_idx  cons_conf_idx  euribor3m  \
0  nonexistent           1.4          93.444          -36.1      4.963
1  nonexistent          -0.1          93.200          -42.0      4.021
2      success          -1.7          94.055          -39.8      0.729
3  nonexistent          -1.8          93.075          -47.1      1.405
4      success          -2.9          92.201          -31.4      0.869

   nr_employed  y
0       5228.1  0
1       5195.8  0
2       4991.6  1
3       5099.1  0
4       5076.2  1
(41188, 21)
['age' 'job' 'marital' 'education' 'default' 'housing' 'loan' 'contact'
 'month' 'day_of_week' 'duration' 'campaign' 'pdays' 'previous' 'poutcome'
 'emp_var_rate' 'cons_price_idx' 'cons_conf_idx' 'euribor3m' 'nr_employed'
 'y']
```


File Edit View Navigate Code Refactor Run Tools VCS Window Help
ITP449_Fall2020 - in_class_coding.py
ITP449_Fall2020 〉 Class 〉 In Class Coding 〉 in_class_coding.py
in_class_coding
1: Project
7: Structure
2: Favorites
Pr...
in_class_coding.py
ITP449_Fall2(
Class
.idea
Files
In Class
in_cl
venv li
External Libr
Scratches and
1 import pandas as pd
2 import os
3 import matplotlib.pyplot as plt
4 import seaborn as sb
5
6 os.chdir('C:/Users/Reza/Desktop/ITP449_Fall2020/Class/Files')
7 pd.set_option('display.max_columns', None)
8
9 bankData = pd.read_csv('banking.csv', header=0)
10
11 bankData.drop(bankData.columns[[0, 3, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19]],
12 axis=1, inplace=True)
13
14 print(bankData.head())
15
2 11 211
Run: in_class_coding
C:\Users\Reza\Desktop\ITP449_Fall2020\Class\venv\Scripts\python.exe "C:/Users/Reza/Desktop/ITP449_Fall2020/Class/In Class Coding/in_cla
job marital default housing loan poutcome y
0 housemaid married no no no nonexistent no
1 services married unknown no no nonexistent no
2 services married no yes no nonexistent no
3 admin. married no no no nonexistent no
4 services married no no yes nonexistent no
4: Run
TODO
6: Problems
Terminal
Python Console
Event Log
17:1 CRLF UTF-8 4 spaces Python 3.8 (Class)

Do the following:

Create dummy variables for all the categorical variables that you have kept.

Age	Status	Height	Income
23	Student	61	Medium
13	Student	55	Medium
36	Unemployed	66	Low
31	Student	64	Low
58	Retired	70	Medium
29	Unemployed	63	High
39	Employed	67	Medium
50	Employed	70	Medium
23	Unemployed	61	Low
36	Employed	66	High

Target attribute

"Dummy" to the rescue!

If using a data analytics technique that requires numeric predictor attributes, how to deal with “Status?”

Here, status is a predictor attribute, but is categorical.

Some techniques require predictor attributes to be categorical.

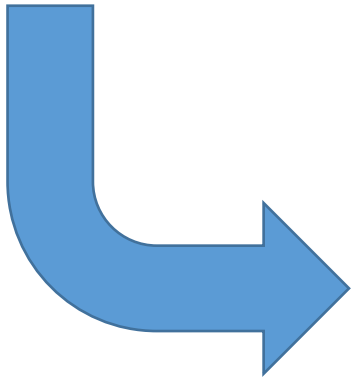
For logistic regression need numerical predictors.

Dummy attributes -- Example

Age	Status	Height	Income
23	Student	61	5000
13	Student	55	1000
36	Unemployed	66	3000
31	Student	64	4000
58	Retired	70	30000
29	Unemployed	63	10000
39	Employed	67	50000
50	Employed	70	55000
23	Unemployed	61	2000
36	Employed	66	20000

Notice how the different values of the categorical variable "Status" have now become attributes with 0-1 values.

But, why only 3?



Age	Student	Unemployed	Employed	Height	Income
23	1	0	0	61	5000
13	1	0	0	55	1000
36	0	1	0	66	3000
31	1	0	0	64	4000
58	0	0	0	70	30000
29	0	0	0	63	10000
39	0	0	1	67	50000
50	0	0	1	70	55000
23	0	1	0	61	2000
36	0	0	1	66	20000

What is the "status" for this row?

Purely nominal variables can be converted into numeric values by introducing 0-1 dummy attributes.
Dummy attributes are simply the possible values of the nominal attribute. We choose n-1 of the values because we can infer the value of the other, based on these.

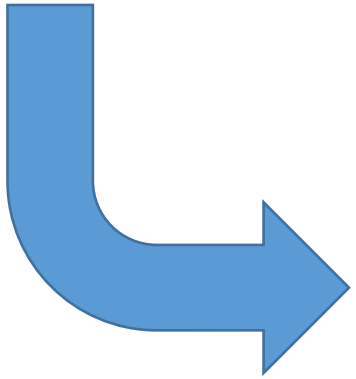
Your Turn

Create dummy attributes for “Gender”

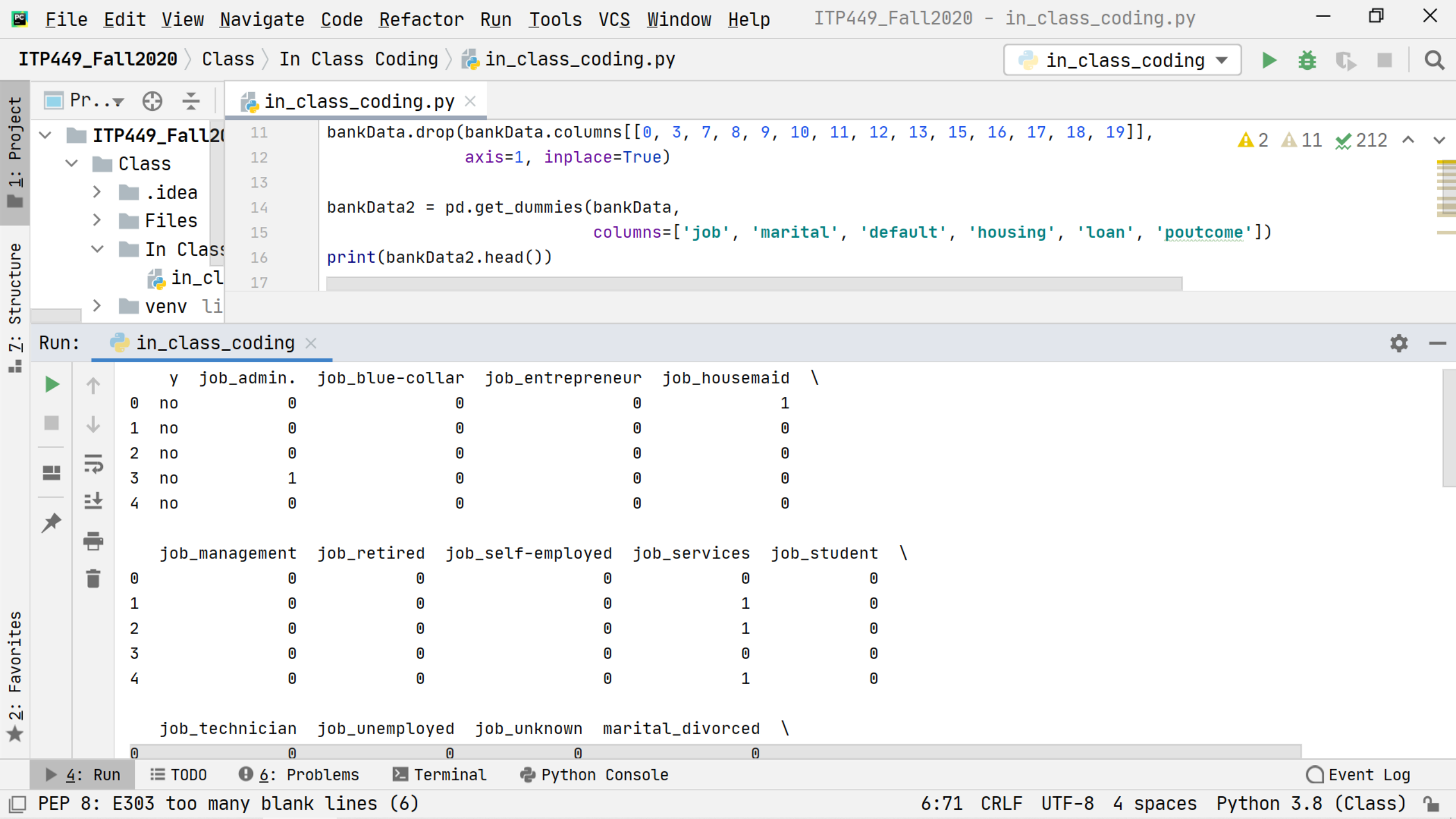
Age	Gender	Height	Income
23	F	61	5000
13	M	55	1000
36	M	66	3000
31	M	64	4000
58	F	70	30000
29	F	63	10000
39	M	67	50000
50	M	70	55000
23	F	61	2000
36	M	66	20000

Answer

Age	Gender	Height	Income
23	F	61	5000
13	M	55	1000
36	M	66	3000
31	M	64	4000
58	F	70	30000
29	F	63	10000
39	M	67	50000
50	M	70	55000
23	F	61	2000
36	M	66	20000



Age	F	Height	Income
23	1	61	5000
13	0	55	1000
36	0	66	3000
31	0	64	4000
58	1	70	30000
29	1	63	10000
39	0	67	50000
50	0	70	55000
23	1	61	2000
36	0	66	20000


File Edit View Navigate Code Refactor Run Tools VCS Window Help
ITP449_Fall2020 - in_class_coding.py
ITP449_Fall2020 〉 Class 〉 In Class Coding 〉 in_class_coding.py
in_class_coding
1: Project
7: Structure
2: Favorites
Pr...
in_class_coding.py
ITP449_Fall2
Class
.idea
Files
In Class
in_cl
venv li
11 bankData.drop(bankData.columns[[0, 3, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19]],
12 axis=1, inplace=True)
13
14 bankData2 = pd.get_dummies(bankData,
15 columns=['job', 'marital', 'default', 'housing', 'loan', 'poutcome'])
16 print(bankData2.head())
17
2 11 212
Run: in_class_coding
y job_admin. job_blue-collar job_entrepreneur job_housemaid \
0 no 0 0 0 1
1 no 0 0 0 0
2 no 0 0 0 0
3 no 1 0 0 0
4 no 0 0 0 0
job_management job_retired job_self-employed job_services job_student \
0 0 0 0 0 0
1 0 0 0 1 0
2 0 0 0 1 0
3 0 0 0 0 0
4 0 0 0 1 0
job_technician job_unemployed job_unknown marital_divorced \
0 0 0 0 0
4: Run
TODO
6: Problems
Terminal
Python Console
Event Log
PEP 8: E303 too many blank lines (6)
6:71 CRLF UTF-8 4 spaces Python 3.8 (Class)

Do the following:

Drop all the *'unknown'* columns.

```
bankData = pd.read_csv('banking.csv', header=0)

bankData.drop(bankData.columns[[0, 3, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19]],
              axis=1, inplace=True)

bankData2 = pd.get_dummies(bankData,
                           columns=['job', 'marital', 'default', 'housing', 'loan', 'poutcome'])

bankData2.drop(bankData2.columns[[12, 16, 18, 21, 24]], axis=1, inplace=True)

print(bankData2.columns)
```

2 11 212

Run: in_class_coding

```
Index(['y', 'job_admin.', 'job_blue-collar', 'job_entrepreneur',
       'job_housemaid', 'job_management', 'job_retired', 'job_self-employed',
       'job_services', 'job_student', 'job_technician', 'job_unemployed',
       'marital_divorced', 'marital_married', 'marital_single', 'default_no',
       'default_yes', 'housing_no', 'housing_yes', 'loan_no', 'loan_yes',
       'poutcome_failure', 'poutcome_nonexistent', 'poutcome_success'],
      dtype='object')
```

Do the following:

Split the data into a training and test set.

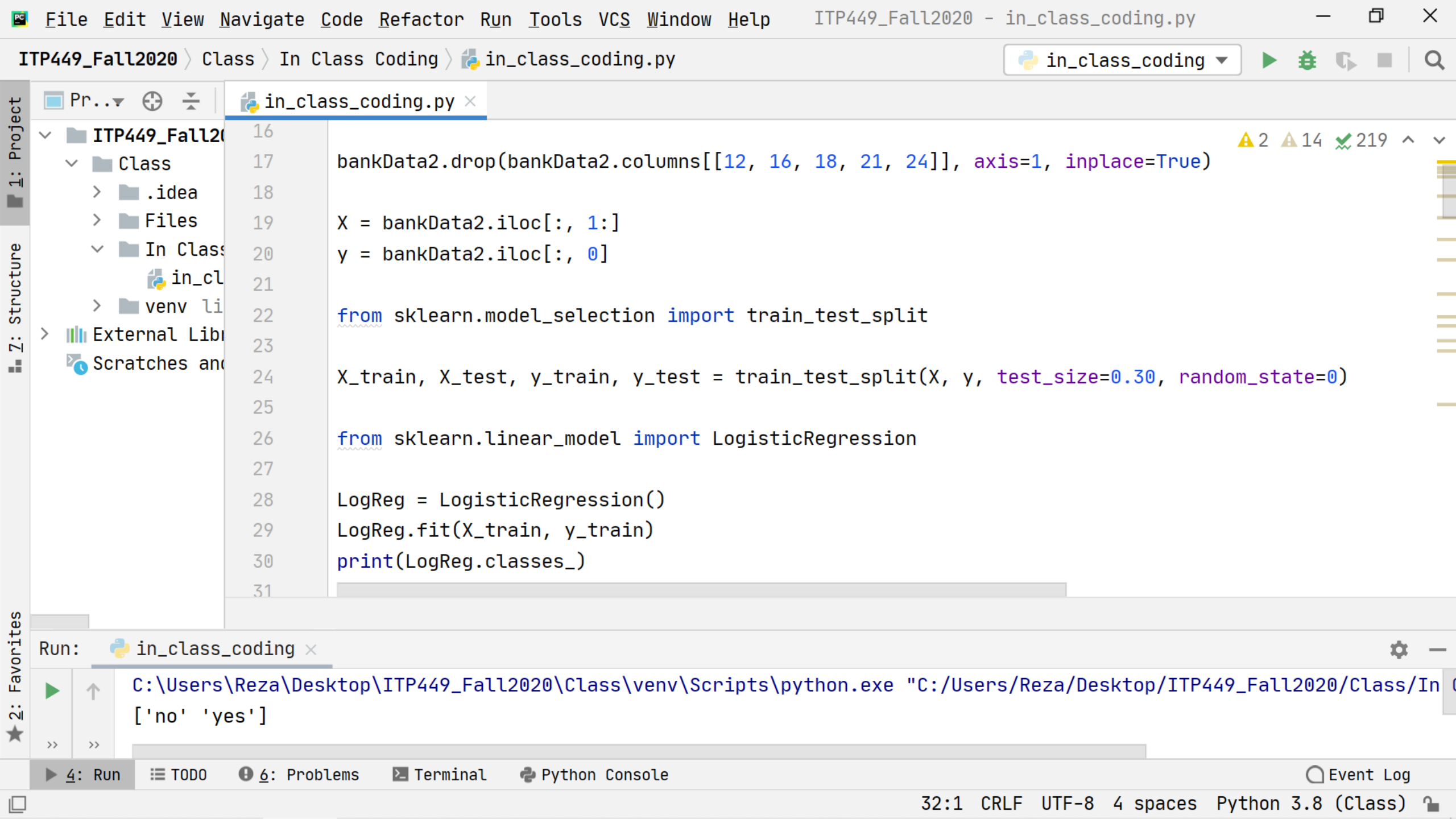
```
17  bankData2.drop(bankData2.columns[[12, 16, 18, 21, 24]], axis=1, inplace=True)
18
19  X = bankData2.iloc[:, 1:]
20  y = bankData2.iloc[:, 0]
21
22  from sklearn.model_selection import train_test_split
23
24  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=0)
25
26  print(X_train.shape)
27  print(y_train.shape)
28  print(X_test.shape)
29  print(y_test.shape)
```


Run: in_class_coding

```
(28831, 23)
(28831,)
(12357, 23)
(12357,)
```

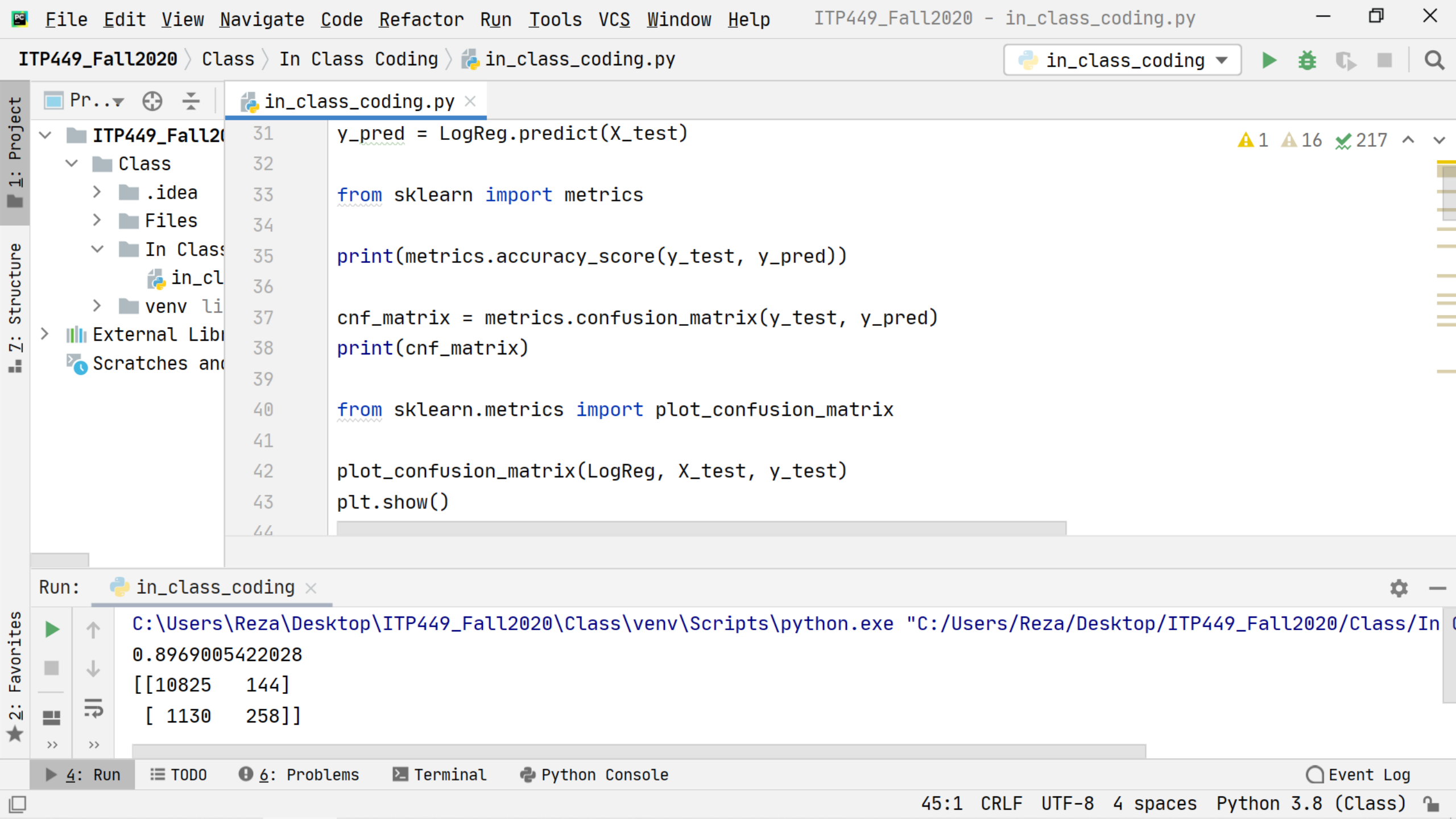
Do the following:

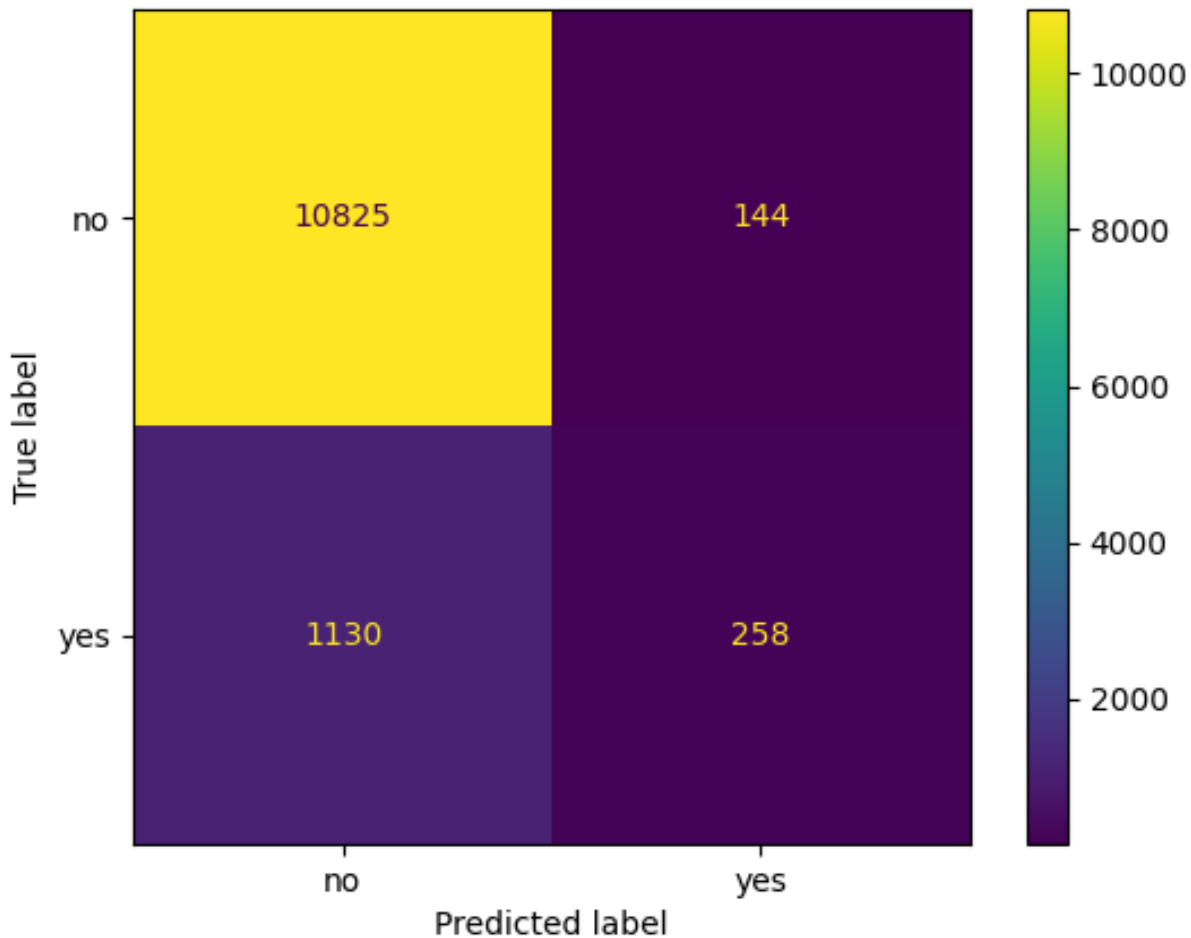
Fit your training data to a Logistic Regression Model

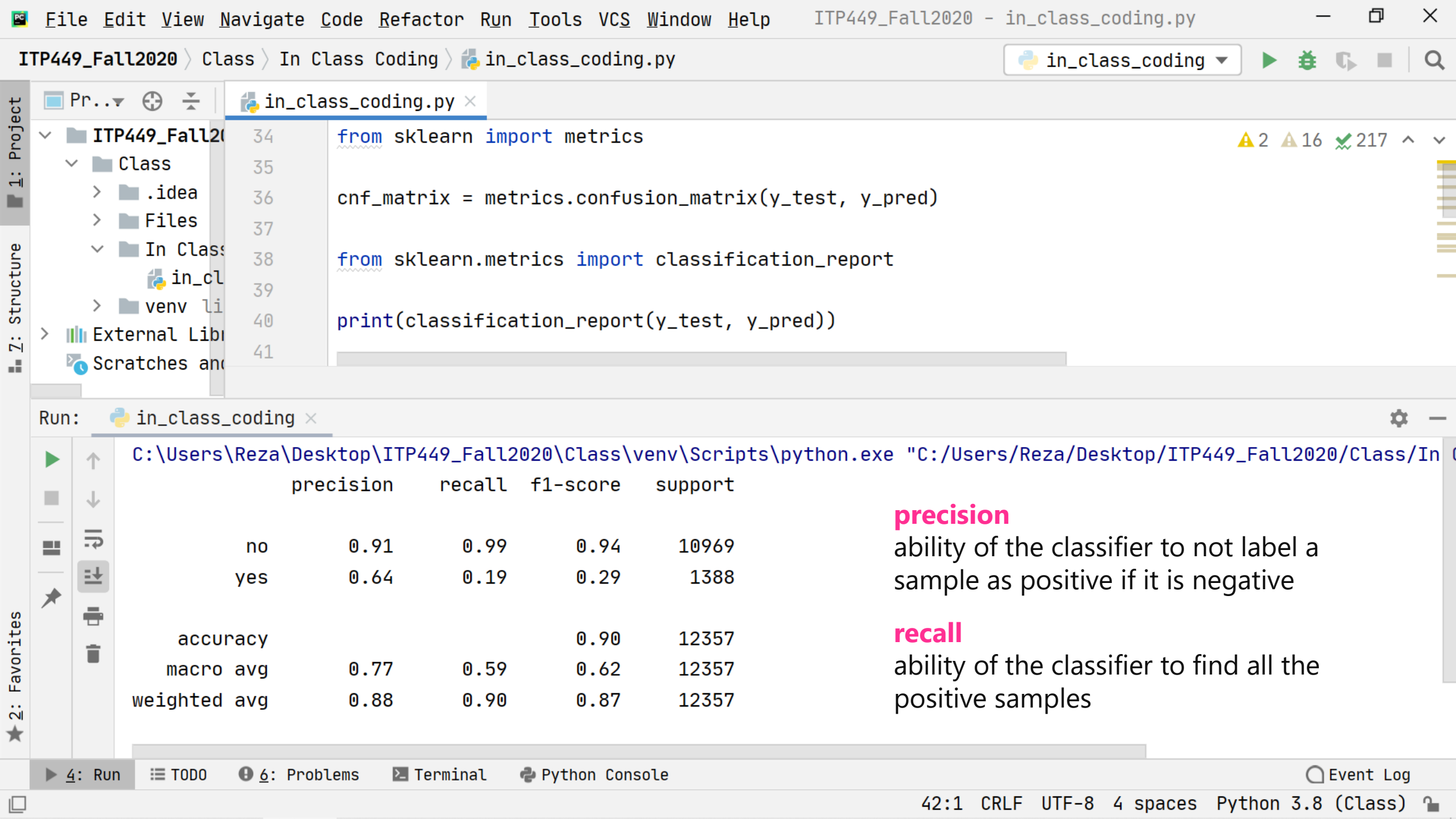

File Edit View Navigate Code Refactor Run Tools VCS Window Help
ITP449_Fall2020 - in_class_coding.py
ITP449_Fall2020 〉 Class 〉 In Class Coding 〉 in_class_coding.py
in_class_coding
Pr..
in_class_coding.py
ITP449_Fall2(
Class
.idea
Files
In Class
in_cl
venv li
External Libr
Scratches and
2 14 219
16
17 bankData2.drop(bankData2.columns[[12, 16, 18, 21, 24]], axis=1, inplace=True)
18
19 X = bankData2.iloc[:, 1:]
20 y = bankData2.iloc[:, 0]
21
22 from sklearn.model_selection import train_test_split
23
24 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=0)
25
26 from sklearn.linear_model import LogisticRegression
27
28 LogReg = LogisticRegression()
29 LogReg.fit(X_train, y_train)
30 print(LogReg.classes_)
31
Run: in_class_coding
C:\Users\Reza\Desktop\ITP449_Fall2020\Class\venv\Scripts\python.exe "C:/Users/Reza/Desktop/ITP449_Fall2020/Class/In
['no' 'yes']
1: Project
7: Structure
2: Favorites
4: Run
TODO
6: Problems
Terminal
Python Console
Event Log
32:1 CRLF UTF-8 4 spaces Python 3.8 (Class)

Do the following:

Use the testing data to make predictions and determine the accuracy of your predictions.


File Edit View Navigate Code Refactor Run Tools VCS Window Help
ITP449_Fall2020 - in_class_coding.py
ITP449_Fall2020 〉 Class 〉 In Class Coding 〉 in_class_coding.py
in_class_coding
1: Project
7: Structure
2: Favorites
in_class_coding.py
31 y_pred = LogReg.predict(X_test)
32
33 from sklearn import metrics
34
35 print(metrics.accuracy_score(y_test, y_pred))
36
37 cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
38 print(cnf_matrix)
39
40 from sklearn.metrics import plot_confusion_matrix
41
42 plot_confusion_matrix(LogReg, X_test, y_test)
43 plt.show()
1 16 217
Run: in_class_coding
C:\Users\Reza\Desktop\ITP449_Fall2020\Class\venv\Scripts\python.exe "C:/Users/Reza/Desktop/ITP449_Fall2020/Class/In
0.8969005422028
[[10825 144]
[1130 258]]
4: Run
TODO
6: Problems
Terminal
Python Console
Event Log
45:1 CRLF UTF-8 4 spaces Python 3.8 (Class)


10825
144
1130
258
no
yes
True label
no
yes
Predicted label
10000
8000
6000
4000
2000


File Edit View Navigate Code Refactor Run Tools VCS Window Help
ITP449_Fall2020 - in_class_coding.py
ITP449_Fall2020 〉 Class 〉 In Class Coding 〉 in_class_coding.py
in_class_coding
1: Project
7: Structure
2: Favorites
in_class_coding.py
ITP449_Fall2
Class
.idea
Files
In Class
in_cl
venv li
External Libr
Scratches and
34 from sklearn import metrics
35
36 cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
37
38 from sklearn.metrics import classification_report
39
40 print(classification_report(y_test, y_pred))
41
2 16 217
Run: in_class_coding
C:\Users\Reza\Desktop\ITP449_Fall2020\Class\venv\Scripts\python.exe "C:/Users/Reza/Desktop/ITP449_Fall2020/Class/In
precision recall f1-score support
no 0.91 0.99 0.94 10969
yes 0.64 0.19 0.29 1388
accuracy 0.90 12357
macro avg 0.77 0.59 0.62 12357
weighted avg 0.88 0.90 0.87 12357
precision
ability of the classifier to not label a sample as positive if it is negative
recall
ability of the classifier to find all the positive samples
4: Run
TODO
6: Problems
Terminal
Python Console
Event Log
42:1 CRLF UTF-8 4 spaces Python 3.8 (Class)

- The **precision** is the ratio *tp / (tp + fp)* where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier to not label a sample as positive if it is negative.

- The **recall** is the ratio *tp / (tp + fn)* where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

- The **F-beta** score can be interpreted as a weighted harmonic mean of the precision and recall, where an F-beta score reaches its best value at 1 and worst score at 0.

  *2 * recall * precision / (recall + precision)*

- The F-beta score weights the recall more than the precision by a factor of beta. beta = 1.0 means recall and precision are equally important.

- The **support** is the number of occurrences of each class in y_test.