Homework 5 Part 2

November 20, 2023

1 Homework 5 Part 2

Due: Monday, November 20, 11:59 PM

This is an individual assignment.

1.1 Description

Create or edit this Jupyter Notebook to answer the questions below. Use simulations to answer these questions. An analytical solution can be useful to check if your simulation is correct but analytical solutions alone will not be accepted as a solution to a problem.

2 Problem 5

A surveillance camera periodically checks a certain area and records a signal $X = W$ if there is no intruder (this is the null hypothesis $H_0$). If there is an intruder the signal is $X = \theta + W$, where $\theta$ is unknown with $\theta > 0$. We assume that $W$ is a normal random variable with mean 0 and known variance $v = 0.6$.

(a) We obtain a single signal value $X = 0.86$. Should $H_0$ be rejected at the 5% level of significance?

(b) We obtain five independent signal values $X = 0.93, -0.36, 0.89, 0.42, -0.27$. Should $H_0$ be rejected at the 5% level of significance?

(c) Repeat part (b), using a t-distribution, and assuming the variance v is unknown.

```
[ ]: import scipy.stats as stats
     import numpy as np
     import matplotlib.pyplot as plt

     '''
     # Problem 5

     A surveillance camera periodically checks a certain area and records a signal␣
      ↪$X=W$ if there is no intruder (this is the null hypothesis $H_0$). If there␣
      ↪is an intruder the signal is $X=\theta+W$, where $\theta$ is unknown with␣
      ↪$\theta>0$. We assume that $W$ is a normal random variable with mean 0 and␣
      ↪known variance $v=0.6$.
```

```
(a) We obtain a single signal value $X=0.86$. Should $H_0$ be rejected at the␣
 ↪5% level of significance?

(b) We obtain five independent signal values $X=0.93, -0.36, 0.89, 0.42, -0.27$.
 ↪ Should $H_0$ be rejected at the 5% level of significance?

(c) Repeat part (b), using a t-distribution, and assuming the variance $v$ is␣
 ↪unknown.
'''
# Part a
mean = 0
var = 0.6
std = np.sqrt(var)

W = stats.norm(mean, std)
ci = W.interval(0.95)
print(f'The upper bound of the CI is {ci[1]:.3f}, thus we cannot reject the␣
 ↪null hypothesis.')

# Part b
X = np.array([0.93, -0.36, 0.89, 0.42, -0.27])
print(f'None of the following values are outside the CI: {X} considering our CI␣
 ↪is {ci}')

# Part c
W_t = stats.t((len(X)-1), loc=mean, scale=std)
W_t_ci = W_t.interval(0.95)
print(f'The new upperbound of the t distribution CI is {W_t_ci[1]:.3f}, thus we␣
 ↪cannot reject the null hypothesis.')
```

```
The upper bound of the CI is 1.518, thus we cannot reject the null hypothesis.
None of the following values are outside the CI: [ 0.93 -0.36  0.89  0.42 -0.27]
considering our CI is (-1.518181574257992, 1.518181574257992)
The new upperbound of the t distribution CI is 2.151, thus we cannot reject the
null hypothesis.
```

3 Problem 6

Leukemia is a type of cancer found in the blood and bone marrow and is caused by the rapid production of abnormal white blood cells (leukocytes). These abnormal white blood cells are not able to fight infection and impair the ability of the bone marrow to produce red blood cells and platelets.

Suppose that leukocytes cell count for people without Leukemia/cancer can be approximately modeled with a Gaussian($\mu = 8, \sigma^2 = 5$) and leukocytes cell count for people with Leukemia/cancer are approximately Gaussian($\mu = 14, \sigma^2 = 6$).

Answer the following questions:

1. (4 points) Determine the threshold γ for the leukocytes cell count such that the probability of a miss is 5%.

```
[ ]: nl_mean = 8
     nl_var = 5

     no_leuk = stats.norm(nl_mean, np.sqrt(nl_var))

     l_mean = 14
     l_var = 6

     leuk = stats.norm(l_mean, np.sqrt(l_var))

     leuk_ci = leuk.interval(0.95)
     miss_prob = 0.05

     gamma = leuk.ppf(miss_prob)
     print(f'The gamma value is {gamma:.3f}')
```

```
The gamma value is 9.971
```

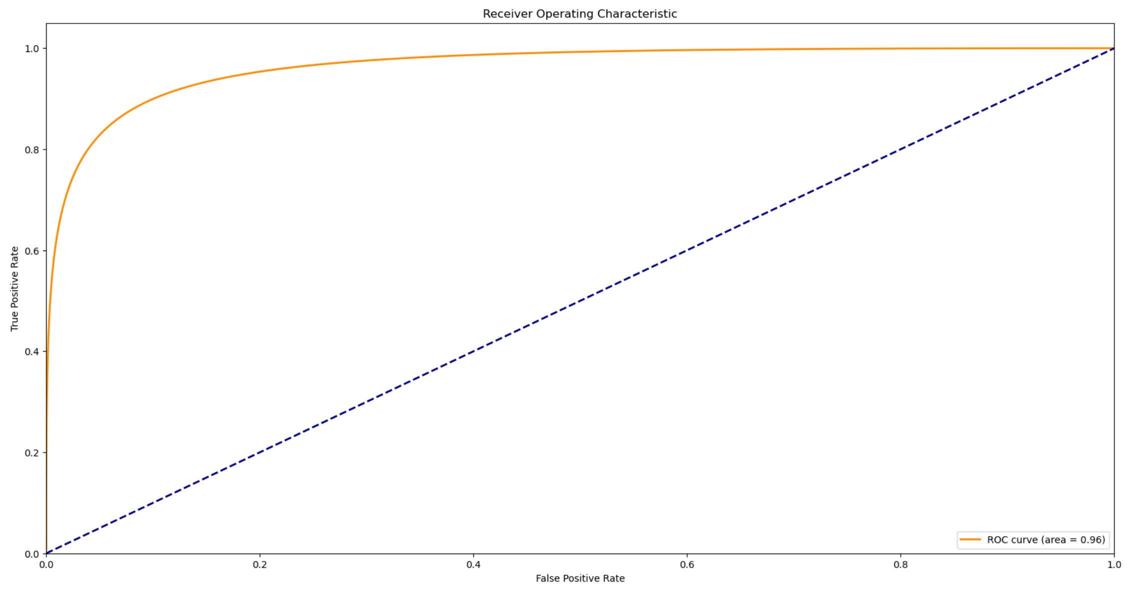
2. (4 points) What is the probability of false alarm using the threshold value γ found in part (1).

```
[ ]: prob_false_alarm = 1 - no_leuk.cdf(gamma)
     print(f"The probability of a false alarm is: {prob_false_alarm:.4f}")
```

```
The probability of a false alarm is: 0.1890
```

3. (4 points) Compute FPR and TPR, plot the ROC curve and compute the AUC.

```
[ ]: from sklearn.metrics import roc_curve, auc
     th = np.linspace(0, 20, 1000)

     FPR = [1 - no_leuk.cdf(gamma) for gamma in th]
     TPR = [1 - leuk.cdf(gamma) for gamma in th]

     roc_auc = auc(FPR, TPR)

     # Plotting the ROC curve

     plt.figure(figsize=(20, 10))
     plt.plot(FPR, TPR, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' %␣
      ↪roc_auc)
     plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
     plt.xlim([0.0, 1.0])
     plt.ylim([0.0, 1.05])
     plt.xlabel('False Positive Rate')
     plt.ylabel('True Positive Rate')
     plt.title('Receiver Operating Characteristic')
     plt.legend(loc='lower right')
     plt.show()
```



4. (3 points) What is the range of value for the threshold γ such that the FPR is less than 15%?

```
[ ]: new_gamma = no_leuk.ppf(0.85)
     print(f'The new lower bound gamma value is {new_gamma:.3f}, reducing the FPR to␣
      ↪only 15%')
```

```
The new lower bound gamma value is 10.318, reducing the FPR to only 15%
```

4 Problem 7

This problem uses the Iris dataset from the UCI ML Repository and available with `scikit-learn`.

Let's load the data:

```
[ ]: from sklearn.datasets import load_iris
     import pandas as pd
     import numpy as np

     iris = load_iris(return_X_y=False)

     print(iris.DESCR)
```

```
.. _iris_dataset:

Iris plants dataset
```

```
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. topic:: References

   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
     Structure and Classification Rule for Recognition in Partially Exposed
     Environments".  IEEE Transactions on Pattern Analysis and Machine
     Intelligence, Vol. PAMI-2, No. 1, 67-71.
   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
     on Information Theory, May 1972, 431-433.
   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
     conceptual clustering system finds 3 classes in the data.
   - Many, many more …
```

This data set contains samples from 3 types of Iris plants: 0 (setosa), 1 (versicolor) and 2 (virginica):

```
[ ]: iris.target_names
```

```
[ ]: array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

```
[ ]: df = pd.DataFrame(data=np.hstack((iris.data, iris.target[:,np.newaxis])),
                       columns=np.hstack((iris.feature_names,'Iris Class')))
     df.head()
```

```
[ ]:    sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
     0                5.1               3.5                1.4               0.2
     1                4.9               3.0                1.4               0.2
     2                4.7               3.2                1.3               0.2
     3                4.6               3.1                1.5               0.2
     4                5.0               3.6                1.4               0.2

        Iris Class
     0         0.0
     1         0.0
     2         0.0
     3         0.0
     4         0.0
```

Let's consider each iris flower to be characterized by its sepal length in cm.

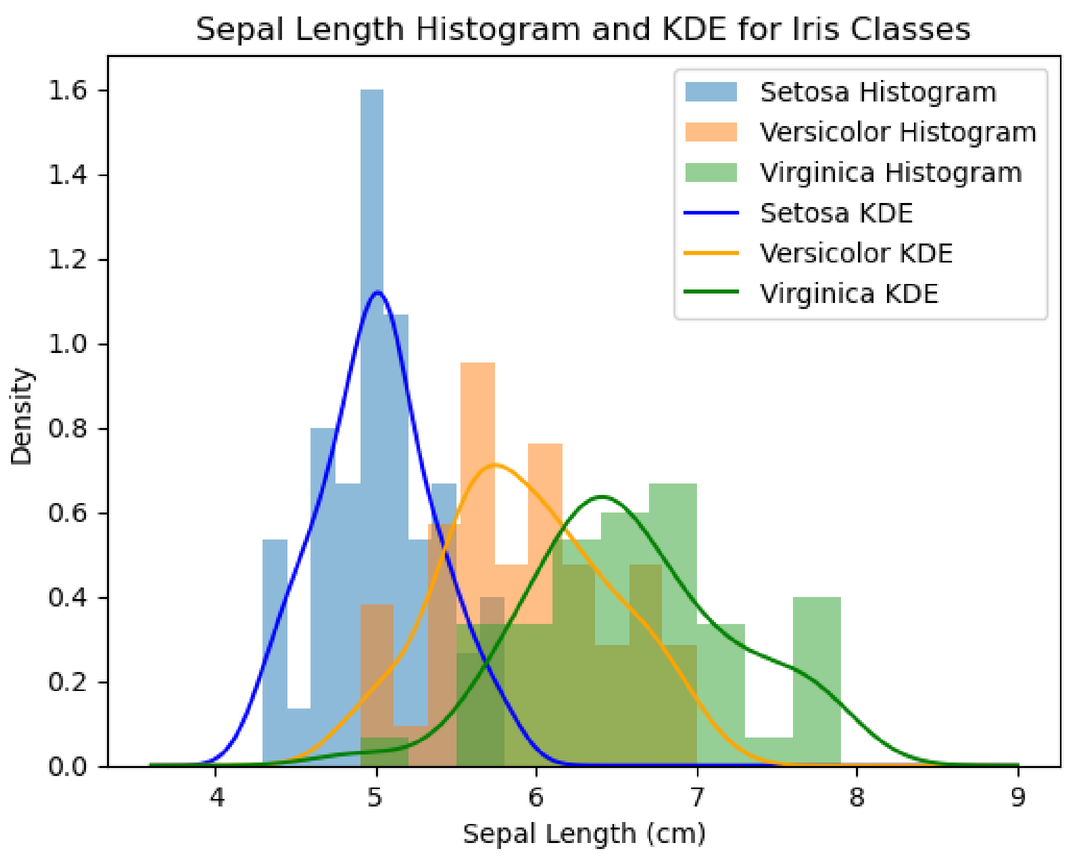
Answer the following questions:

1. Plot the histogram for the three types of iris flowers. Overlay the 3 histograms, add a legend and axis labels.
2. Estimate the density function for each type of iris using Kernel Density Estimation (KDE) with a Gaussian kernel.
3. In the same plot, plot the Gaussian KDE for each class along with each class histogram. Include legends and axis labels.
4. Compute the prior probability for each type of iris based on this data.
5. Find the region of values (you may use `x=np.linspace(3.6,9,1000)`) for which MAP will decide iris class 0 (setosa), 1 (versicolor) and 2 (virginica).

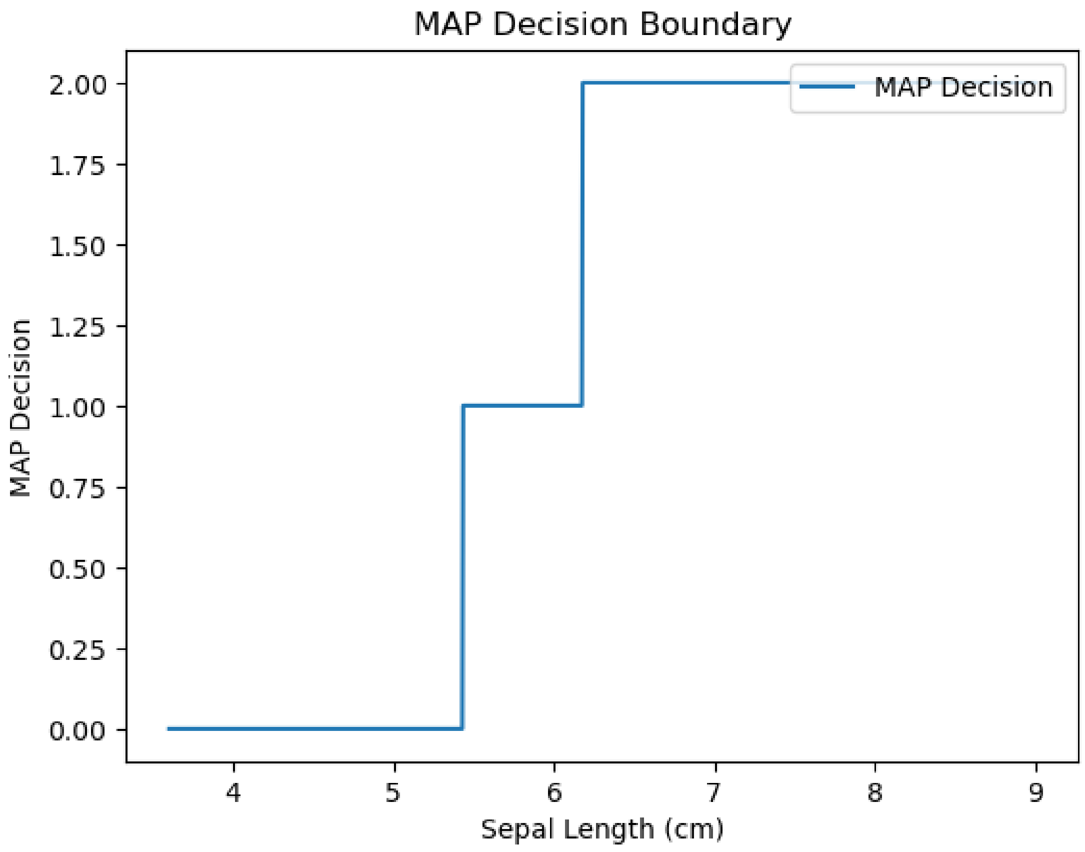
6. Based on this decision rule, compute how many samples will be falsely classified as class 0 (setosa).

```
[ ]: # 1-3.
     # find min and max of sepal length
     min_max = df['sepal length (cm)'].agg(['min', 'max'])
     x = np.linspace(3.6, 9, 1000)

     kde_setosa = stats.gaussian_kde(df[df['Iris Class'] == 0]['sepal length (cm)'])
     kde_versicolor = stats.gaussian_kde(df[df['Iris Class'] == 1]['sepal length␣
      ↪(cm)'])
     kde_virginica = stats.gaussian_kde(df[df['Iris Class'] == 2]['sepal length␣
      ↪(cm)'])

     plt.hist(df[df['Iris Class'] == 0]['sepal length (cm)'], bins=10, alpha=0.5,␣
      ↪density=True, label='Setosa Histogram')
     plt.hist(df[df['Iris Class'] == 1]['sepal length (cm)'], bins=10, alpha=0.5,␣
      ↪density=True, label='Versicolor Histogram')
     plt.hist(df[df['Iris Class'] == 2]['sepal length (cm)'], bins=10, alpha=0.5,␣
      ↪density=True, label='Virginica Histogram')
     plt.plot(x, kde_setosa(x), label='Setosa KDE',color='blue')
     plt.plot(x, kde_versicolor(x), label='Versicolor KDE',color='orange')
     plt.plot(x, kde_virginica(x), label='Virginica KDE',color='green')
     plt.xlabel('Sepal Length (cm)')
     plt.ylabel('Density')
     plt.title('Sepal Length Histogram and KDE for Iris Classes')
     plt.legend(loc='upper right')
     plt.show()

     # 4.
     prior_setosa = len(df[df['Iris Class'] == 0]) / len(df)
     prior_versicolor = len(df[df['Iris Class'] == 1]) / len(df)
     prior_virginica = len(df[df['Iris Class'] == 2]) / len(df)

     print(f'The prior probability of Setosa is {prior_setosa:.3f}, the prior␣
      ↪probability of Versicolor is {prior_versicolor:.3f}, and the prior␣
      ↪probability of Virginica is {prior_virginica:.3f}')

     # 5. Find the region of values (you may use ```x=np.linspace(3.6,9,1000)```)␣
      ↪for which MAP will decide iris class 0 (setosa), 1 (versicolor) and 2␣
      ↪(virginica).
     map_classes = np.zeros_like(x)
     for i in range(len(x)):
         posterior_setosa = prior_setosa * kde_setosa(x[i])
         posterior_versicolor = prior_versicolor * kde_versicolor(x[i])
         posterior_virginica = prior_virginica * kde_virginica(x[i])
```

```
    map_classes[i] = np.argmax([posterior_setosa, posterior_versicolor,␣
↪posterior_virginica])

plt.plot(x, map_classes, label='MAP Decision')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('MAP Decision')
plt.title('MAP Decision Boundary')
plt.legend(loc='upper right')
plt.show()

# 6. Based on this decision rule, compute how many samples will be falsely␣
↪classified as class 0 (setosa).
critical_point = None
for x_i in x:
    if kde_setosa(x_i) < kde_versicolor(x_i) or kde_setosa(x_i) <␣
↪kde_virginica(x_i):
        critical_point = x_i
        break
critical_point
setosa_df = df[df['Iris Class'] == 0]
missed_setosa = len(setosa_df[setosa_df['sepal length (cm)'] > critical_point])
print(f'The number of samples falsely classified as not class 0 (setosa) that␣
↪are setosa\'s is {missed_setosa}')
```



```
The prior probability of Setosa is 0.333, the prior probability of Versicolor is
0.333, and the prior probability of Virginica is 0.333
```

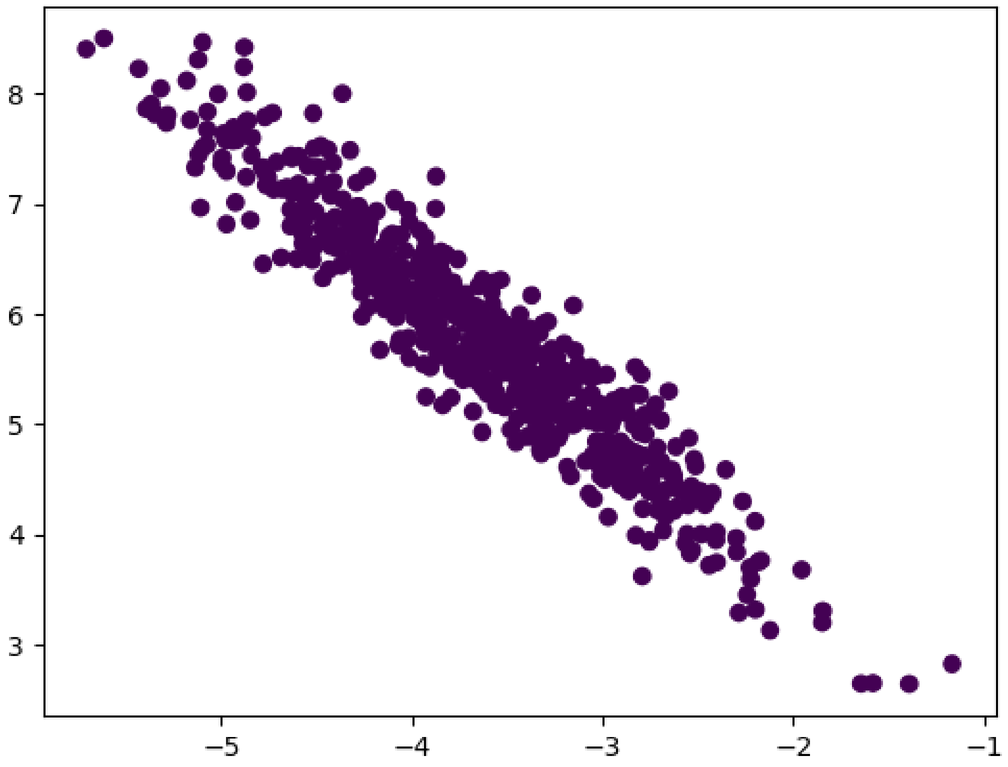


```
The number of samples falsely classified as not class 0 (setosa) that are
setosa's is 5
```

5 Problem 8

Consider the following dataset:

```
[ ]: from sklearn.datasets import make_blobs

     X, y = make_blobs(n_samples=600,centers=1)
     X = np.dot(X, [[0.60834549, -0.63667341], [-0.40887718, 0.85253229]])
     plt.scatter(X[:,0],X[:,1],c=y);
```



Answering the following questions:

1. Compute the covariance matrix.
2. Compute the correlation coefficient using the formula.
3. Find the OLS solution using x to predict y, what is the MSE and R^2 of the OLS solution?
4. Plot the fitted linear function with the orginal dataset.
5. Performe the Null Hypothesis Tests for Correlation to decide whether the dependence between x and y with significance level of 95%.

[ ]:
```
X_df = pd.DataFrame(data=X, columns=['x1', 'x2'])
# 1. Compute the covariance matrix.
X_df.cov()
```

[ ]:
```
          x1        x2
x1  0.547858 -0.737854
x2 -0.737854  1.113462
```

[ ]:
```
# 2. Compute the correlation coefficient using the formula.
# the formula is cov(X, Y) / (std(X) * std(Y))
cov_matrix = X_df.cov()
```

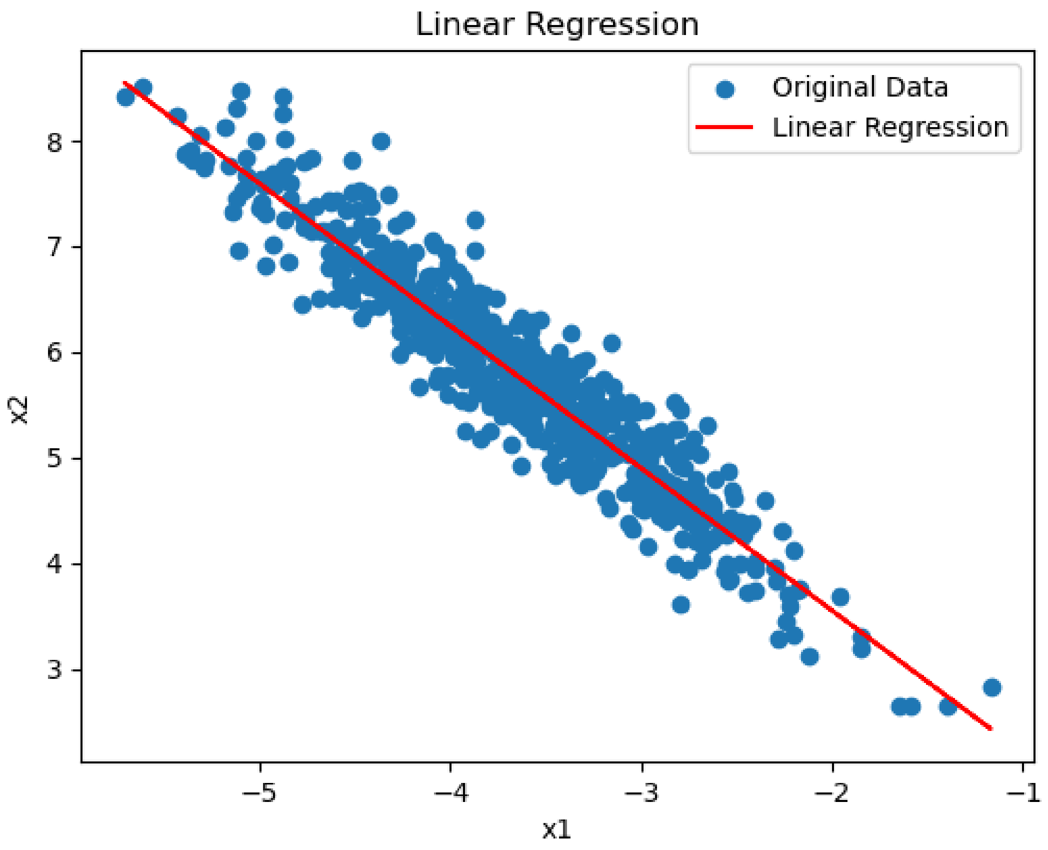
```
corr = cov_matrix.iloc[0, 1] / (np.sqrt(cov_matrix.iloc[0, 0]) * np.
 ↪sqrt(cov_matrix.iloc[1, 1]))
print(f'The correlation coefficient is {corr:.4f}')
```

```
The correlation coefficient is -0.9447
```

```
[ ]: # 3. Find the OLS solution using $x$ to predict $y$, what is the MSE and $R^2$␣
      ↪of the OLS solution?
     regressor = stats.linregress(X_df['x1'], X_df['x2'])

     errors = X_df['x2'] - ( regressor.intercept + regressor.slope * X_df['x1'])
     print(f'The MSE of the OLS solution is {np.mean(errors**2):.4f}')
     print(f'The R^2 of the OLS solution is {regressor.rvalue**2:.4f}')
```

```
The MSE of the OLS solution is 0.1195
The R^2 of the OLS solution is 0.8925
```

```
[ ]: # 4. Plot the fitted linear function with the orginal dataset.
     plt.scatter(X_df['x1'], X_df['x2'], label='Original Data')
     plt.plot(X_df['x1'], regressor.intercept + regressor.slope * X_df['x1'],␣
      ↪color='red', label='Linear Regression')
     plt.xlabel('x1')
     plt.ylabel('x2')
     plt.title('Linear Regression')
     plt.legend(loc='upper right')
     plt.show()
```



```
[ ]: # 5. Performe the Null Hypothesis Tests for Correlation to decide whether the␣
       ↪dependence between $x$ and $y$ with significance level of 95%.
     r , p_value = stats.pearsonr(X_df['x1'], X_df['x2'])

     print(f'The p-value is {p_value:.4f}, thus we cannot reject the null hypothesis␣
       ↪that there is no correlation between x1 and x2')
```

```
The p-value is 0.0000, thus we cannot reject the null hypothesis that there is
no correlation between x1 and x2
```

---

6 Submission Instructions:

When you are done with the exercises in this notebook, upload a PDF or your results to Canvas. To create the PDF with your code and results, you can use the following procedure:

1. Go to Kernel
2. Click Restart and Run All
3. Check over the notebook to make sure everything still looks right

At this point, you may be able to just choose “Print” from JupyterLab’s File menu and then print to PDF (OS dependent). If everything is correct in the PDF version, then upload that PDF to the assignment in Canvas.

If your PDF is missing any of your outputs, you can use the following procedure:

4. Next, click File at the top on the tool bar below Jupyter icon
5. Click Save and Export Notebook as… and choose HTML
6. The HTML file will either open in a new tab/window or be downloaded to your Downloads folder. Open it if it is in the Downloads folder
7. Print the HTML file to PDF (how to do this is OS dependent). Make sure to save it to somewhere you can find it
8. Open the PDF to make sure that everything looks right and that nothing is cut off
9. Upload both the PDF and ipynb files to the Canvas assignment