Open in Colab

Homework 3

In this homework assignment, we will further explore the **k-nearest-neighbor (kNN)** model for classification and the **linear regression** model for regression tasks.

Please submit your solutions as a PDF file to Blackboard by **Wednesday, March 16th at 11:59 PM**.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn
from sklearn.linear_model import LinearRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn import datasets
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
%matplotlib inline
```

Part 1: kNN Method

Evaluate the kNN method with $k = 1, 5, 25$ on the dataset used in Week 3 notebook.

```
url = "https://raw.githubusercontent.com/empathy87/The-Elements-of-Statistical-Learning-
data = pd.read_csv(url)
model_svm = LinearRegression()
input_col = ['x1', 'x2']
data.head()
```

	x1	x2	y
0	2.526093	0.321050	0
1	0.366954	0.031462	0
2	0.768219	0.717486	0
3	0.693436	0.777194	0
4	-0.019837	0.867254	0

1. Split the data into 80% training data and 20% test data.

```
1 training_data, test_data = train_test_split(data, test_size=0.2)
2 test_data
```

2. Train a kNN model with $k = 1$ on the training data, and calculate its prediction accuracy on the test data.

```
model_1nn = KNeighborsClassifier(n_neighbors=1)
model_1nn.fit(data[['x1', 'x2']], data['y'])
data['prediction'] = model_1nn.predict(data[input_col])
accuracy_score(data['y'], data['prediction'])
```

```
1.0
```

3. Train a kNN model with $k = 5$ on the training data, and calculate its prediction accuracy on the test data.

```
model_5nn = KNeighborsClassifier(n_neighbors=5)
model_5nn.fit(data[['x1', 'x2']], data['y'])
data['prediction_5'] = model_5nn.predict(data[input_col])
accuracy_score(data['y'], data['prediction_5'])
```

```
0.87
```

4. Train a kNN model with $k = 50$ on the training data, and calculate its prediction accuracy on the test data.

```
model_50nn = KNeighborsClassifier(n_neighbors=50)
model_50nn.fit(data[['x1', 'x2']], data['y'])
data['prediction_50'] = model_50nn.predict(data[input_col])
accuracy_score(data['y'], data['prediction_50'])
```

```
0.81
```

5. Which k value give the best accuracy score? Can you explain why?

Answer: 1 gave the best K value because it is just assigned to the class of the nearest neighbor.

Part 2: Linear Regression

Train a linear regression model using the normal equation and verify that the results coincide with the results from `sklearn`.

We will use the advertisement revenue data used in Week 5 and build a linear regression model on `radio` and `sales`.

```
url = "https://www.statlearning.com/s/Advertising.csv"
advertising = pd.read_csv(url, index_col=0)
advertising.head()
```

1. Split the data into 85% training data and 15% test data.

```
train, test = train_test_split(advertising, test_size=0.15)
```

2. Build a linear regression model that predicts `sales` with the `radio` feature using `sklearn`. Display the model parameters and its test MSE.

```
model = LinearRegression()
model.fit(advertising[['radio']], advertising[['sales']])
m = model.coef_[0, 0]
b = model.intercept_[0]
plt.plot(advertising.radio, advertising.sales, 'r.')
x_coord = np.array([0, 50])
y_coord = x_coord * m + b
plt.plot(x_coord, y_coord, 'b-')
```

```
# MSE
beta0 = m
beta1 = b
y_pred = beta0 + beta1 + advertising.radio
mean_squared_error(advertising.sales, y_pred)
```

```
509.3601299362396
```

3. Calculate the parameter values with the normal equation. These values should be the same as the values shown in Step 2.

```
X = np.hstack([np.ones([len(advertising), 1]), advertising[['radio']].values])
y = advertising[['sales']].values
beta = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(y)
beta
```

```
array([[9.3116381 ],
       [0.20249578]])
```

```
def get_squared_error(beta0, beta1, data, i):
    xi = data.loc[i, 'radio']
    yi = data.loc[i, 'sales']
```

```
    f_xi = beta0 + beta1 * xi
    return (yi - f_xi) ** 2

def get_MSE(beta0, beta1, data):
    list_errors = [get_squared_error(beta0, beta1, data, ind) for ind in data.index]
    return sum(list_errors) / len(list_errors)

get_MSE(9.3116381, 0.20249578, advertising)
```

```
18.09239774512544
```

```
plt.plot(advertising['radio'], advertising['sales'], 'r.')
x_coordinates = np.array([0,50])
y_coordinates = x_coordinates * 0.202 + 9.311
plt.plot(x_coordinates, y_coordinates, 'b-')
```