Machine Learning HW #1 answer

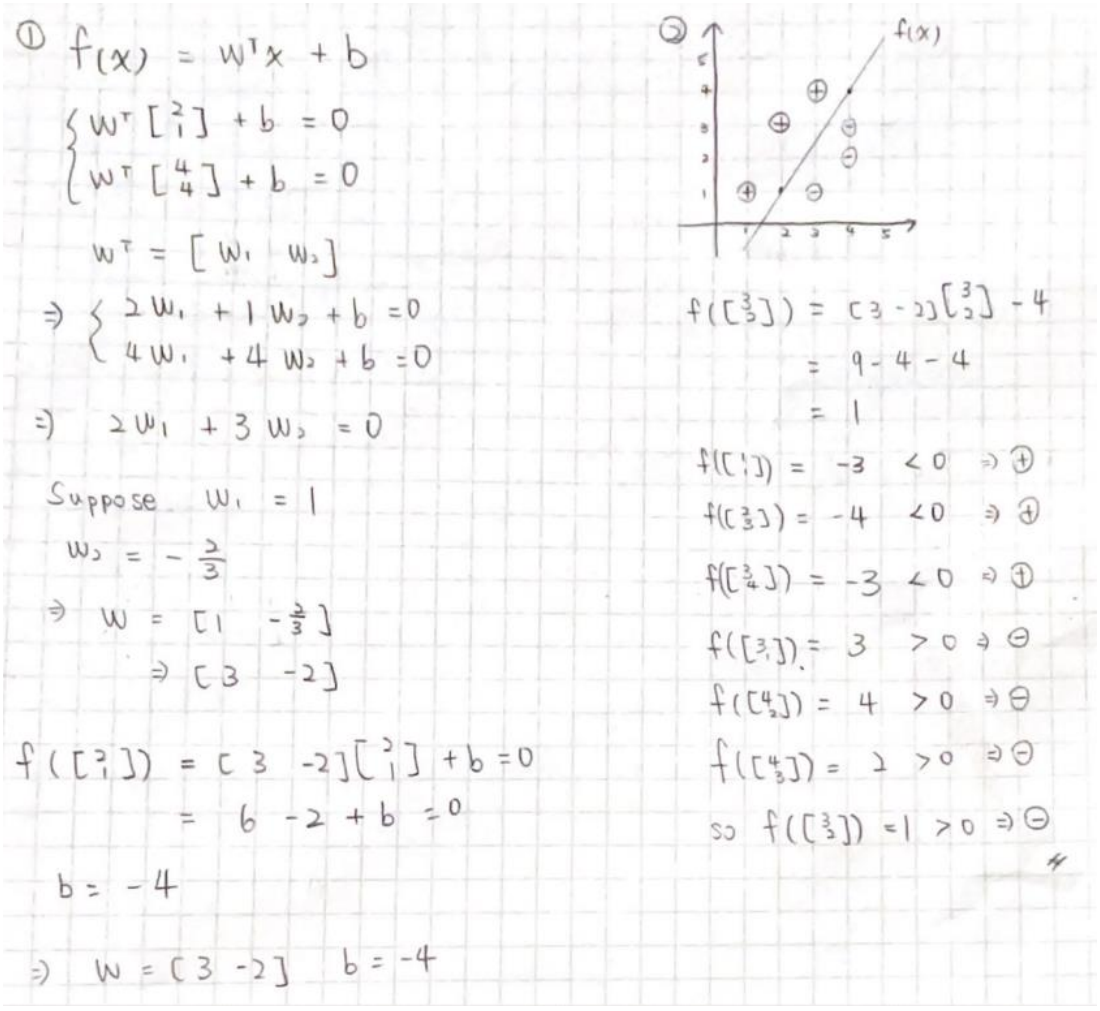
1.

The Turing test is an exam that tests a machine's ability to display intelligent behavior and is considered to be a strong indicator of AI. ChatGPT was able to mimic human-like responses and convince the human evaluators. So, the answer is yes, ChatGPT did pass the Turing test.

2.

- Unsupervised-learning algorithm is for unlabeled data, so it's suitable to use supervised-learning in this case.
- Classification model is the task of predicting a discrete class label.
  Regression model is the task of predicting a continuous quantity.
  We use regression techniques to perform the salary of a college graduate.

3.



4.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

iris_dataset = datasets.load_iris()

df_X = iris_dataset.data[:,:4]
df_y = iris_dataset.target
```

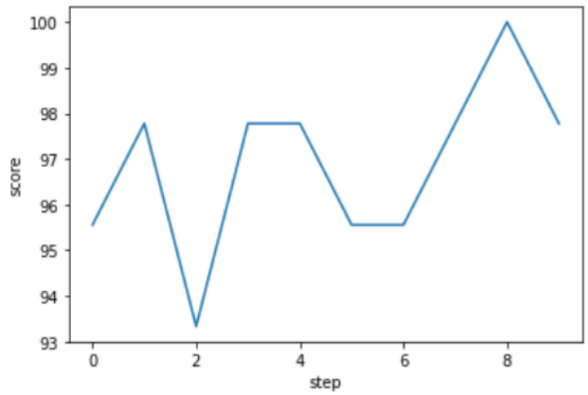
```
scores = []

for i in range(10):
        X_train, X_test, y_train, y_test = train_test_split(df_X, df_y, test_size=0.3)

        classifier = KNeighborsClassifier(n_neighbors=7)
        classifier.fit(X_train, y_train)
        scores.append(round(accuracy_score(classifier.predict(X_test), y_test) * 100, 3))

plt.plot(scores)
plt.xlabel('step')
plt.ylabel('score')
plt.show()

print(f"Score = ", scores)
print(f"Score average: {np.mean(scores)}")
```




```
Score =  [95.556, 97.778, 93.333, 97.778, 97.778, 95.556, 95.556, 97.778, 100.0, 97.778]
Score average: 96.88910000000001
```

5.

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import operator

iris_dataset = datasets.load_iris()
df_X = iris_dataset.data[:, :4]
df_y = iris_dataset.target
```

```
train_size = int(len(df_X)*0.6)
validate_size = int(len(df_X)*0.2)
test_size = len(df_X) - train_size - validate_size
train_X = np.empty((10, train_size, 4))
test_X = np.empty((10, test_size, 4))
train_y = np.empty((10, train_size))
test_y = np.empty((10, test_size))
k_scores = {3:[], 4:[], 5:[], 6:[], 7:[], 8:[], 9:[], 10:[], 11:[]}
scores = {3:[], 4:[], 5:[], 6:[], 7:[], 8:[], 9:[], 10:[], 11:[]}
```

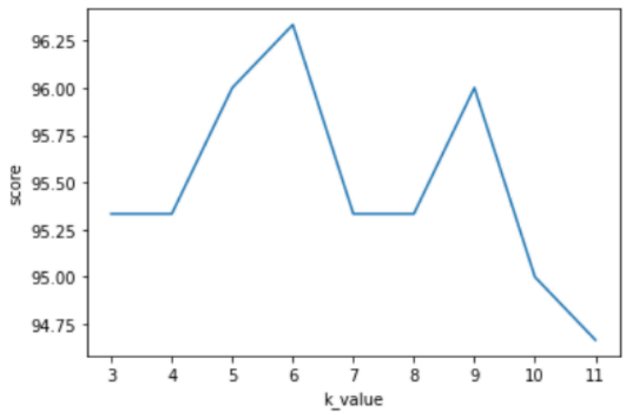
```
for i in range(10):
    X_train, X_test, y_train, y_test = train_test_split(df_X, df_y, test_size=0.2, stratify=df_y)
    X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.25, stratify=y_train)
    train_X[i, :, :] = X_train
    test_X[i, :, :] = X_test
    train_y[i, :] = y_train
    test_y[i, :] = y_test
    for k in range(3, 12):
        classifier = KNeighborsClassifier(n_neighbors=k)
        classifier.fit(X_train, y_train)
        k_scores[k].append(accuracy_score(classifier.predict(X_val), y_val) * 100)

for k in range(3, 12):
    scores[k] = np.mean(k_scores[k])

plt.plot(list(scores.keys()), list(scores.values()))
plt.xlabel('k_value')
plt.ylabel('score')
plt.show()
```

```
best_k = (max(scores.items(), key=operator.itemgetter(1))[0])
score_list = []
for i in range(10):
    classifier = KNeighborsClassifier(n_neighbors=best_k)
    classifier.fit(train_X[i, :, :], train_y[i, :])
    score_list.append(round(accuracy_score(classifier.predict(test_X[i, :, :]), test_y[i, :]) * 100, 3))

print(f"The best K: ", (best_k))
print(f"Score=", score_list)
print(f"Average=", round(np.mean(score_list), 3))
```




```
The best K:  6
Score= [100.0, 100.0, 100.0, 100.0, 100.0, 96.667, 96.667, 96.667, 96.667, 100.0]
Average= 98.667
```