2306aml133-kritika-assignment16

October 14, 2023

[2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[44]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
import seaborn as sns
```

[4]:
```
#Loading the data
df = pd.read_csv('/archive (4).zip')
```

[39]:
```
df.head()
```

[39]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	Pass
0	female	group B	bachelor's degree	standard	none	72	72	74	1
1	female	group C	some college	standard	completed	69	90	88	1
2	female	group B	master's degree	standard	none	90	95	93	1
3	male	group A	associate's degree	free/reduced	none	47	57	44	0
4	male	group C	some college	standard	none	76	78	75	1

[40]:
```
print(df.shape)
```

```
(1000, 9)
```

[41]:
```
df.describe()
```

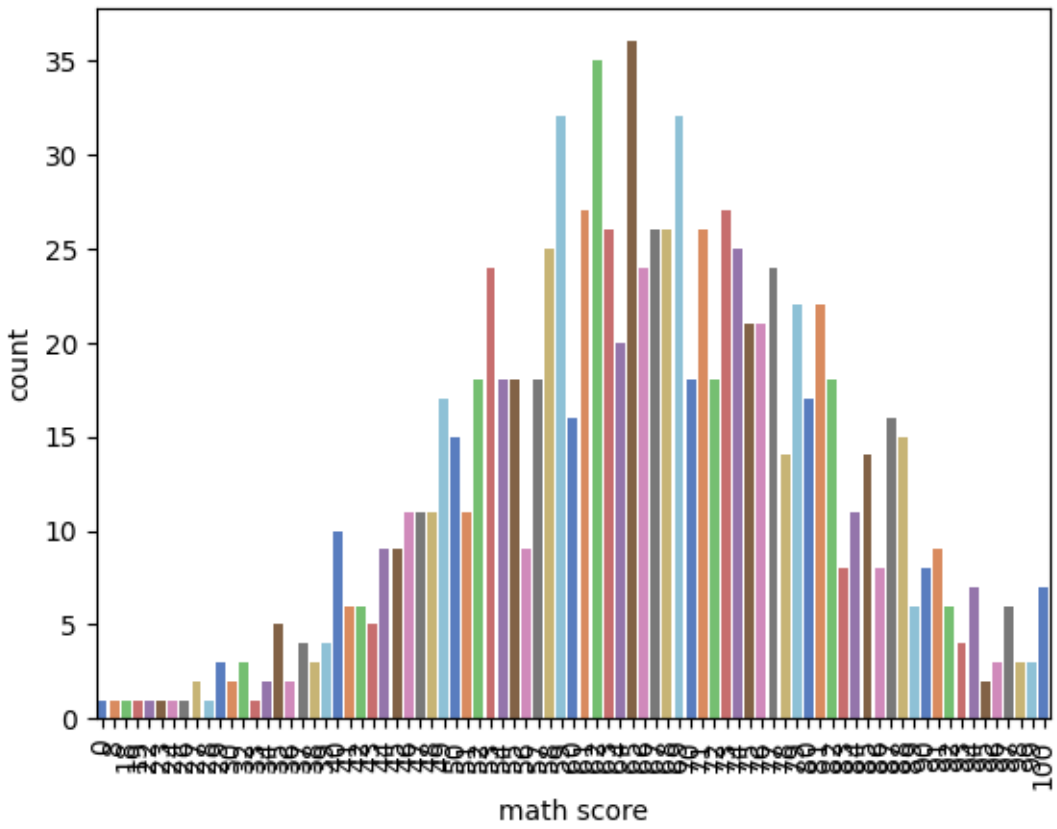
[41]:

	math score	reading score	writing score	Pass
count	1000.00000	1000.000000	1000.000000	1000.00000
mean	66.08900	69.169000	68.054000	0.85000
std	15.16308	14.600192	15.195657	0.35725
min	0.00000	17.000000	10.000000	0.00000
25%	57.00000	59.000000	57.750000	1.00000
50%	66.00000	70.000000	69.000000	1.00000
75%	77.00000	79.000000	79.000000	1.00000
max	100.00000	100.000000	100.000000	1.00000

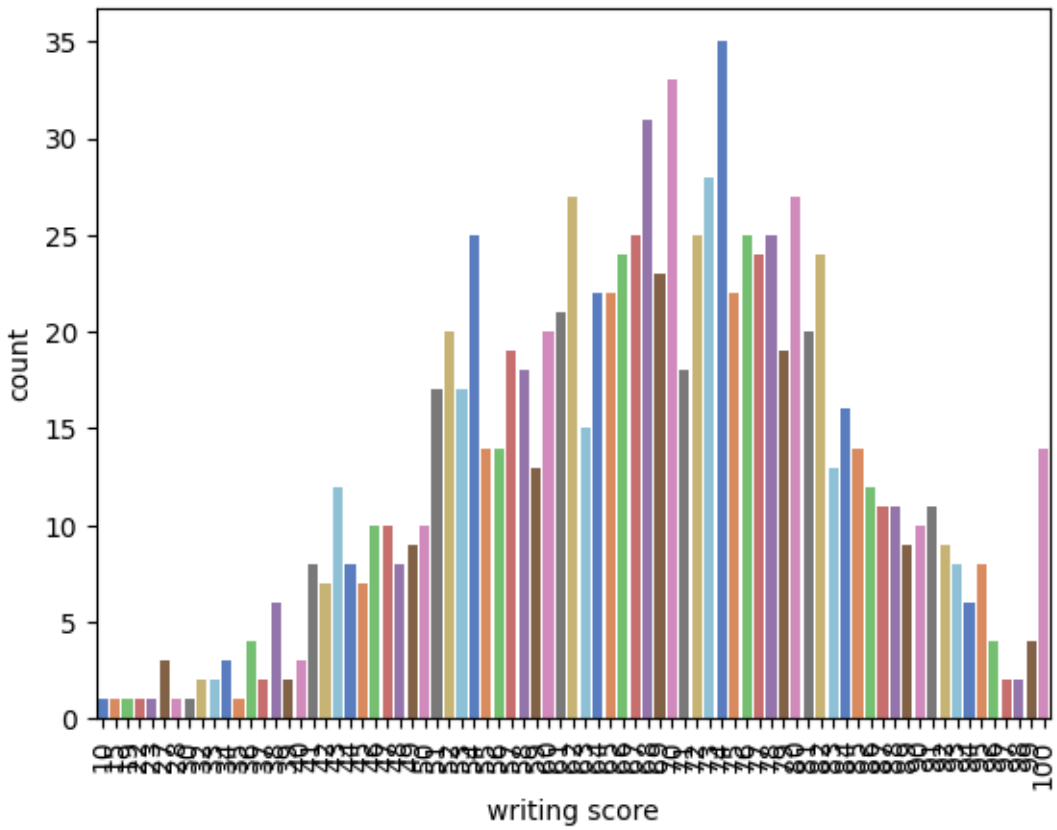
[42]:
```
df.isnull().sum()
```

[42]:
```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
Pass                           0
dtype: int64
```

[50]:
```
#Predicting math score
p = sns.countplot(x="math score", data = df, palette="muted")
_ = plt.setp(p.get_xticklabels(), rotation=90)
```



[51]:
```
#predicting writing score
p = sns.countplot(x="writing score", data = df, palette="muted")
_ = plt.setp(p.get_xticklabels(), rotation=90)
```



[6]:
```
#To check if student is pass or fail by checking their math score
df['Pass'] = np.where(df['math score'] > 50, 1, 0)
```

[7]:
```
X = df[['reading score', 'writing score']]
y = df['Pass']
```

[8]:
```
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

[9]:
```
#Training and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[26]:
```
model_binary = Sequential([
    Dense(64, activation='relu', input_dim=X_train.shape[1]),
    Dropout(0.3),
    Dense(32, activation='sigmoid'),
    Dense(1, activation='sigmoid')
])
```

[29]:
```
model_binary.compile(optimizer=Adam(), loss='binary_crossentropy',␣
↪metrics=['accuracy'])
```

[30]:
```
#Training the data
model_binary.fit(X_train, y_train, epochs=10, batch_size=32,␣
↪validation_data=(X_test, y_test))
```

```
Epoch 1/10
25/25 [==============================] - 1s 10ms/step - loss: -34.7124 -
accuracy: 0.0000e+00 - val_loss: -56.8833 - val_accuracy: 0.0000e+00
Epoch 2/10
25/25 [==============================] - 0s 3ms/step - loss: -84.3708 -
accuracy: 0.0000e+00 - val_loss: -107.6133 - val_accuracy: 0.0000e+00
Epoch 3/10
25/25 [==============================] - 0s 4ms/step - loss: -142.0271 -
accuracy: 0.0000e+00 - val_loss: -168.2955 - val_accuracy: 0.0000e+00
Epoch 4/10
25/25 [==============================] - 0s 4ms/step - loss: -208.1285 -
accuracy: 0.0000e+00 - val_loss: -236.1498 - val_accuracy: 0.0000e+00
Epoch 5/10
25/25 [==============================] - 0s 4ms/step - loss: -279.2769 -
accuracy: 0.0000e+00 - val_loss: -306.3394 - val_accuracy: 0.0000e+00
Epoch 6/10
25/25 [==============================] - 0s 4ms/step - loss: -349.4103 -
accuracy: 0.0000e+00 - val_loss: -373.5433 - val_accuracy: 0.0000e+00
Epoch 7/10
25/25 [==============================] - 0s 4ms/step - loss: -415.7367 -
accuracy: 0.0000e+00 - val_loss: -434.7763 - val_accuracy: 0.0000e+00
Epoch 8/10
25/25 [==============================] - 0s 3ms/step - loss: -476.0924 -
accuracy: 0.0000e+00 - val_loss: -490.5721 - val_accuracy: 0.0000e+00
Epoch 9/10
25/25 [==============================] - 0s 3ms/step - loss: -530.8444 -
accuracy: 0.0000e+00 - val_loss: -542.5214 - val_accuracy: 0.0000e+00
Epoch 10/10
25/25 [==============================] - 0s 4ms/step - loss: -583.6735 -
accuracy: 0.0000e+00 - val_loss: -592.4197 - val_accuracy: 0.0000e+00
```

[30]:
```
<keras.src.callbacks.History at 0x7a86f2d22590>
```

[31]:
```
binary_classification_loss, binary_classification_accuracy = model_binary.
↪evaluate(X_test, y_test)
print('Binary Classification Accuracy:', binary_classification_accuracy)
```

```
7/7 [==============================] - 0s 2ms/step - loss: -592.4197 - accuracy:
0.0000e+00
Binary Classification Accuracy: 0.0
```

[32]:
```
#Predicting the data
predictions_binary = model_binary.predict(X_test)
```

```
7/7 [==============================] - 0s 2ms/step
```

[16]:
```
X = df[['reading score', 'writing score']]
y = df['math score']
```

[17]:
```
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

[18]:
```
#Traning and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[33]:
```
model_regression = Sequential([
    Dense(64, activation='relu', input_dim=X_train.shape[1]),
    Dropout(0.3),  # Dropout layer with a rate of 0.3 (you can experiment with␣
↪different rates)
    Dense(32, activation='relu'),
    Dense(1)  # No activation for regression
])
```

[34]:
```
model_regression.compile(optimizer=Adam(), loss='mean_squared_error',␣
↪metrics=['mean_absolute_error'])
```

[35]:
```
#Training the model
model_regression.fit(X_train, y_train, epochs=10, batch_size=32,␣
↪validation_data=(X_test, y_test))
```

```
Epoch 1/10
25/25 [==============================] - 1s 13ms/step - loss: 4595.8120 -
mean_absolute_error: 66.0976 - val_loss: 4288.9146 - val_mean_absolute_error:
63.5799
Epoch 2/10
25/25 [==============================] - 0s 5ms/step - loss: 4443.3447 -
mean_absolute_error: 64.9248 - val_loss: 4096.4995 - val_mean_absolute_error:
61.9893
Epoch 3/10
25/25 [==============================] - 0s 6ms/step - loss: 4170.7163 -
mean_absolute_error: 62.7342 - val_loss: 3749.9902 - val_mean_absolute_error:
58.9903
Epoch 4/10
25/25 [==============================] - 0s 13ms/step - loss: 3695.5132 -
mean_absolute_error: 58.6825 - val_loss: 3206.7666 - val_mean_absolute_error:
53.9369
Epoch 5/10
```

```
25/25 [==============================] - 0s 9ms/step - loss: 3006.7664 -
mean_absolute_error: 52.2318 - val_loss: 2501.6655 - val_mean_absolute_error:
47.0157
Epoch 6/10
25/25 [==============================] - 0s 11ms/step - loss: 2193.5107 -
mean_absolute_error: 43.8102 - val_loss: 1807.9147 - val_mean_absolute_error:
38.9387
Epoch 7/10
25/25 [==============================] - 1s 22ms/step - loss: 1459.0402 -
mean_absolute_error: 34.7911 - val_loss: 1247.0438 - val_mean_absolute_error:
31.1497
Epoch 8/10
25/25 [==============================] - 0s 9ms/step - loss: 931.4998 -
mean_absolute_error: 26.7667 - val_loss: 844.5955 - val_mean_absolute_error:
24.2565
Epoch 9/10
25/25 [==============================] - 0s 5ms/step - loss: 646.0402 -
mean_absolute_error: 21.3027 - val_loss: 615.8796 - val_mean_absolute_error:
19.8095
Epoch 10/10
25/25 [==============================] - 0s 5ms/step - loss: 500.1060 -
mean_absolute_error: 18.5616 - val_loss: 487.9074 - val_mean_absolute_error:
17.2482
```

[35]:
```
<keras.src.callbacks.History at 0x7a87000c94b0>
```

[36]:
```
regression_loss, regression_mean_absolute_error = model_regression.
  ↪evaluate(X_test, y_test)
print('Regression Mean Absolute Error:', regression_mean_absolute_error)
```

```
7/7 [==============================] - 0s 3ms/step - loss: 487.9074 -
mean_absolute_error: 17.2482
Regression Mean Absolute Error: 17.248205184936523
```

[49]:
```
predictions_regression = model_regression.predict(X_test)
```

```
7/7 [==============================] - 0s 3ms/step
```