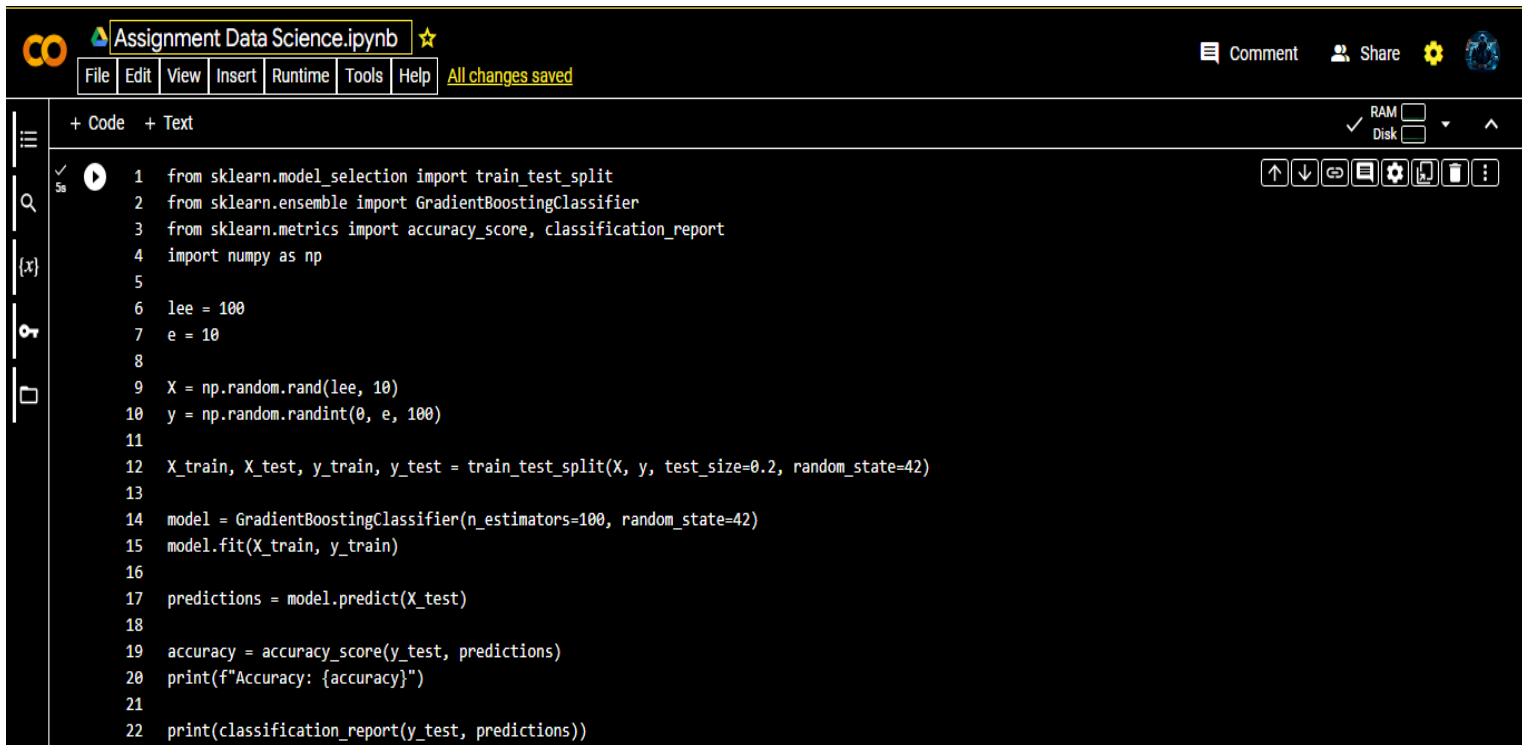


DATA SCIENCE CAT 2

Scenario 1



The screenshot shows a Jupyter Notebook titled "Assignment Data Science.ipynb". The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a status bar indicating "All changes saved". The code is written in a dark-themed editor and consists of 22 lines. It imports necessary libraries, generates random data, splits it into training and testing sets, trains a Gradient Boosting Classifier, and prints the accuracy and classification report.

```
1 from sklearn.model_selection import train_test_split
2 from sklearn.ensemble import GradientBoostingClassifier
3 from sklearn.metrics import accuracy_score, classification_report
4 import numpy as np
5
6 lee = 100
7 e = 10
8
9 X = np.random.rand(lee, 10)
10 y = np.random.randint(0, e, 100)
11
12 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
13
14 model = GradientBoostingClassifier(n_estimators=100, random_state=42)
15 model.fit(X_train, y_train)
16
17 predictions = model.predict(X_test)
18
19 accuracy = accuracy_score(y_test, predictions)
20 print(f"Accuracy: {accuracy}")
21
22 print(classification_report(y_test, predictions))
```

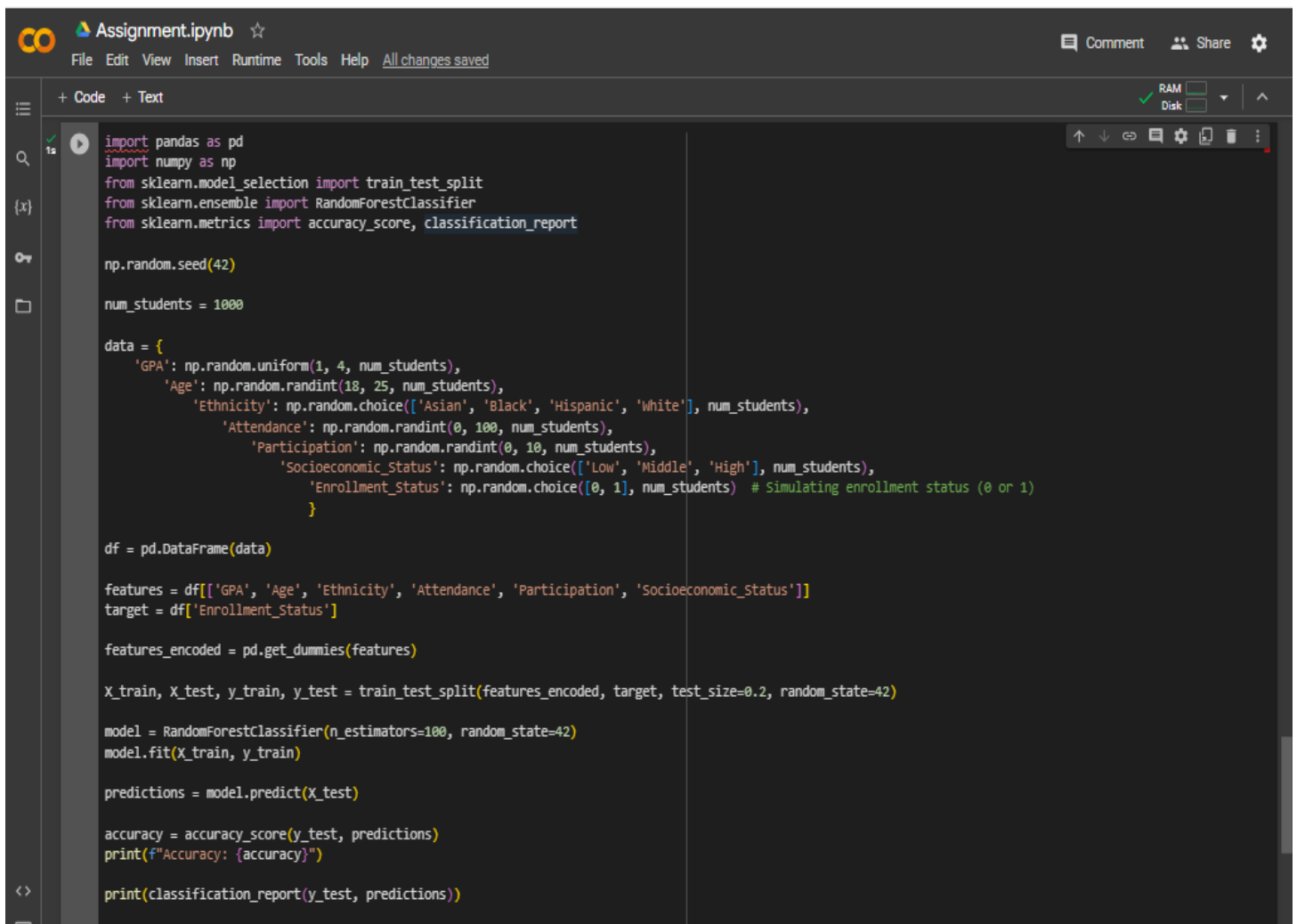
Code

Accuracy: 0.2

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.50 | 0.25 | 0.33 | 4 |
| 1 | 0.00 | 0.00 | 0.00 | 0 |
| 2 | 0.00 | 0.00 | 0.00 | 2 |
| 3 | 0.00 | 0.00 | 0.00 | 2 |
| 4 | 0.00 | 0.00 | 0.00 | 1 |
| 5 | 0.00 | 0.00 | 0.00 | 0 |
| 6 | 0.33 | 0.25 | 0.29 | 4 |
| 7 | 0.00 | 0.00 | 0.00 | 2 |
| 8 | 0.17 | 0.50 | 0.25 | 2 |
| 9 | 0.33 | 0.33 | 0.33 | 3 |
| accuracy | | | 0.20 | 20 |
| macro avg | 0.13 | 0.13 | 0.12 | 20 |
| weighted avg | 0.23 | 0.20 | 0.20 | 20 |

Output

Scenario 2



```
Assignment.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
RAM
Disk

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

np.random.seed(42)

num_students = 1000

data = {
    'GPA': np.random.uniform(1, 4, num_students),
    'Age': np.random.randint(18, 25, num_students),
    'Ethnicity': np.random.choice(['Asian', 'Black', 'Hispanic', 'White'], num_students),
    'Attendance': np.random.randint(0, 100, num_students),
    'Participation': np.random.randint(0, 10, num_students),
    'Socioeconomic_Status': np.random.choice(['Low', 'Middle', 'High'], num_students),
    'Enrollment_Status': np.random.choice([0, 1], num_students) # Simulating enrollment status (0 or 1)
}

df = pd.DataFrame(data)

features = df[['GPA', 'Age', 'Ethnicity', 'Attendance', 'Participation', 'Socioeconomic_Status']]
target = df['Enrollment_Status']

features_encoded = pd.get_dummies(features)

X_train, X_test, y_train, y_test = train_test_split(features_encoded, target, test_size=0.2, random_state=42)

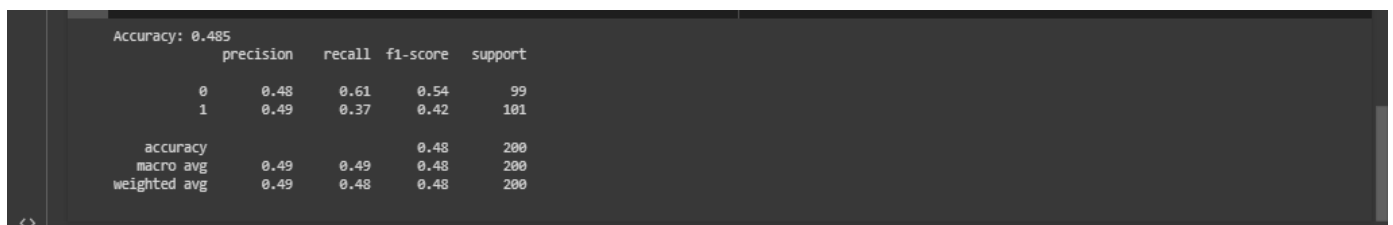
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

predictions = model.predict(X_test)

accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy}")

print(classification_report(y_test, predictions))
```

Code



```
Accuracy: 0.485
precision recall f1-score support
0 0.48 0.61 0.54 99
1 0.49 0.37 0.42 101

accuracy 0.48 200
macro avg 0.49 0.49 0.48 200
weighted avg 0.49 0.48 0.48 200
```

Output

Scenario 3

```
Assignment.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
Insert code cell below Ctrl+MB
(x)
Ov
RAM
Disk

numpy as np
matplotlib.pyplot as plt
tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split

np.random.seed(42)
num_samples = 1000
num_features = 5

data = {'feature_{i}': np.random.rand(num_samples) for i in range(num_features)}
data['equipment_failure'] = np.random.randint(0, 2, num_samples)

df = pd.DataFrame(data)

X = df.drop('equipment_failure', axis=1)
y = df['equipment_failure']

scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(LSTM(units=50))
model.add(Dense(units=1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

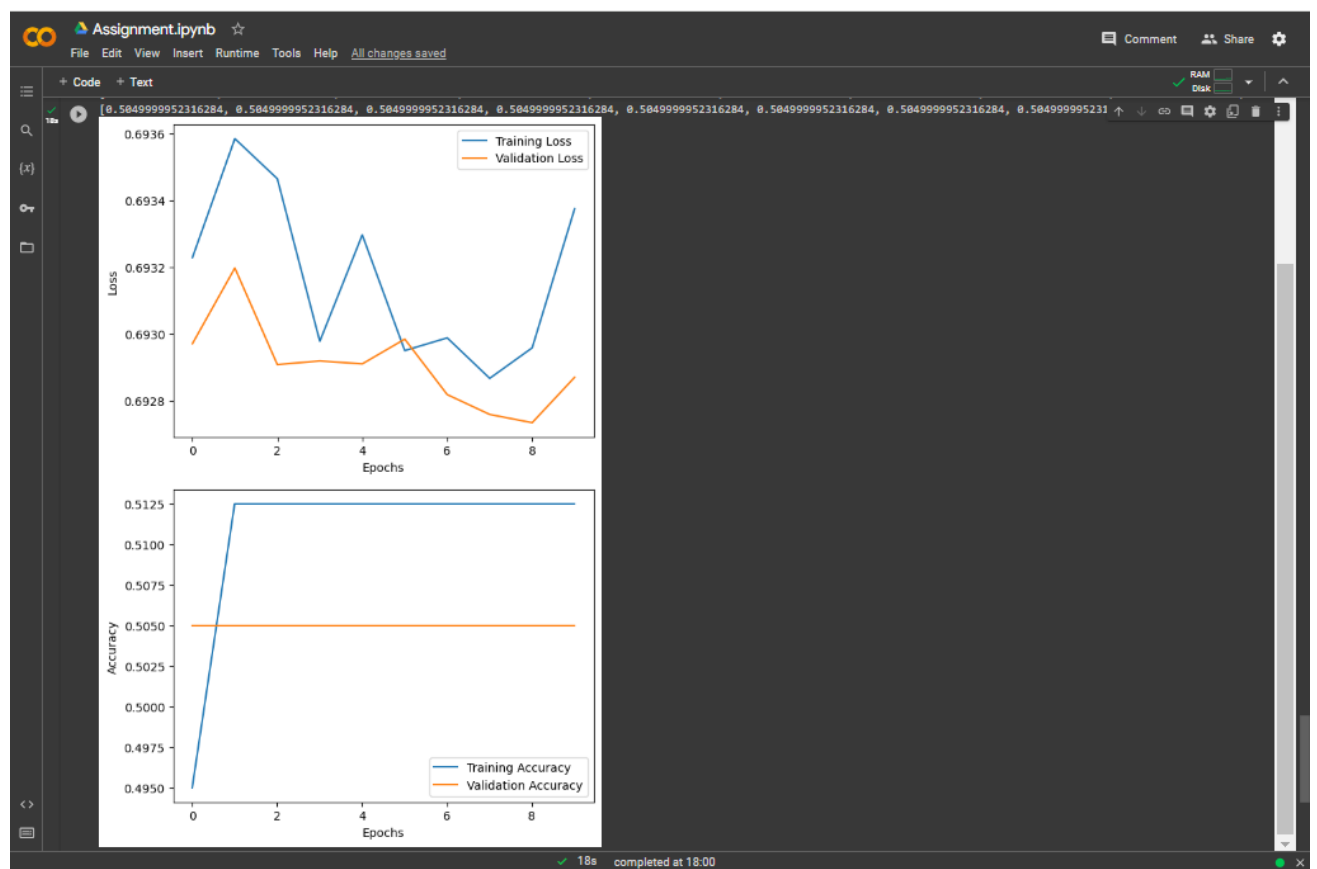
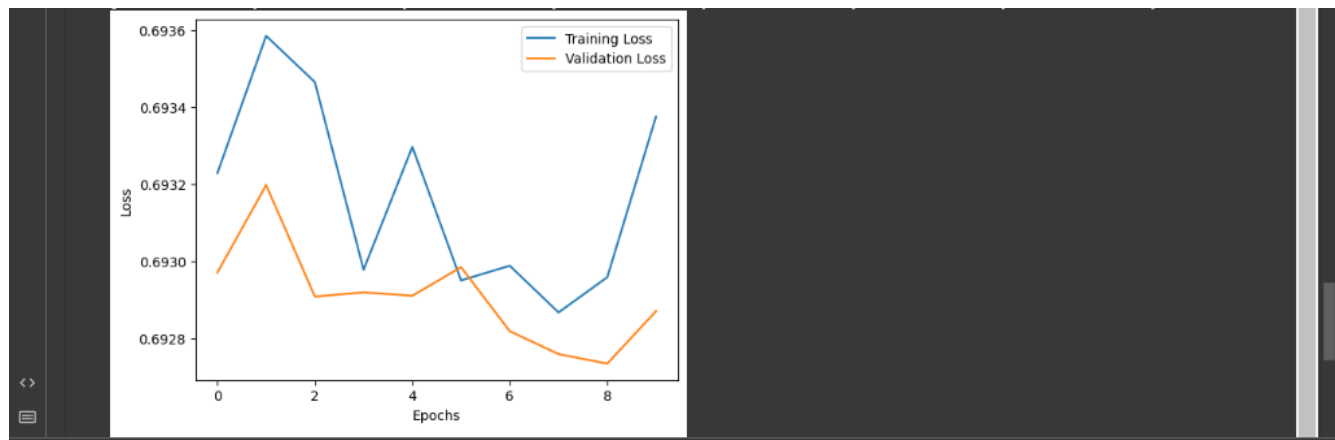
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test), verbose=1)

print(history.history['loss'])
print(history.history['accuracy'])
print(history.history['val_loss'])
print(history.history['val_accuracy'])

plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

Code



Output

GITHUB LINK: <https://github.com/TrevorBrian430/Data-Science>