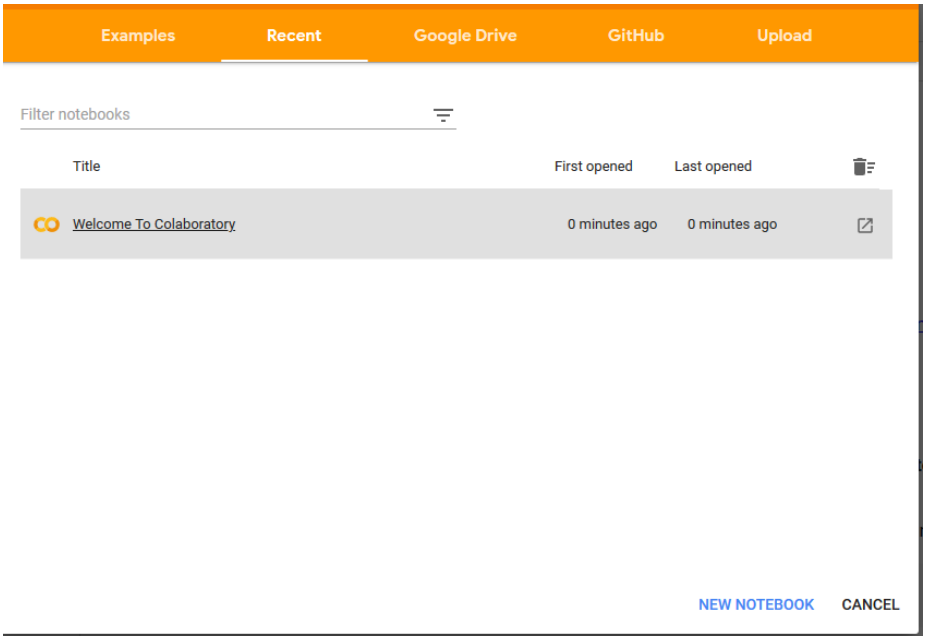
Uygulama 8. Bulut Çalışma Ortamlarının (Google Colab) Kullanımı

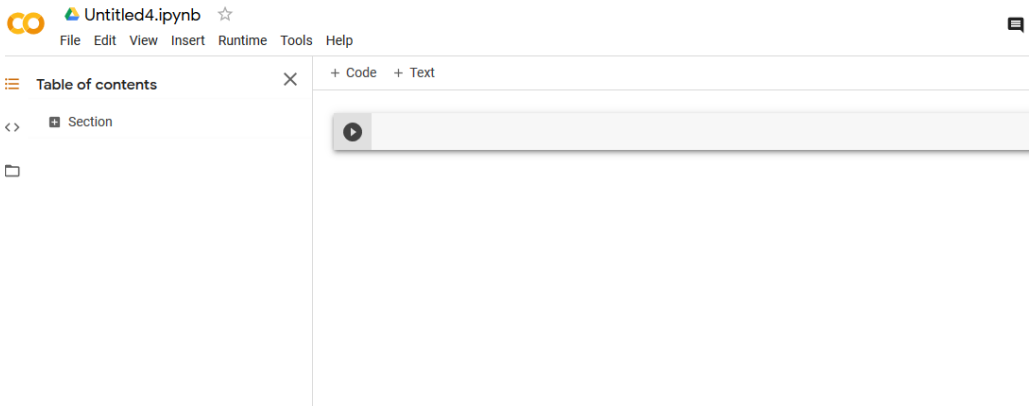
Veri sayısının artması ile artık günümüzde GPU ile yapılan işlemler bile yavaş gerçekleşebilmektedir. Çoğu zaman kişisel bilgisayarlarımızın gücü bu işlemleri yerine getirememekte ya da çok uzun süren saatler boyunca analiz yapmaktadır. Bundan dolayı hem zaman kaybı yaşanmakta hem de analiz işlemi karmaşık olmaktadır. Bu tarz problemlerden kurtulmak için bulut tabanlı çalışma ortamları etkin bir şekilde kullanılmaktadır. En popüler olanı Google Colab'tır. Bu bulut ortamı ile Tesla K80 GPU üzerinde keras işlemleri hızlı ve etkin bir şekilde yapılabilmektir. Bu uygulama çalışmasında, Google Colab'ın kullanımı gösterilecektir.

Google Colab sitesine https://colab.research.google.com/notebooks/intro.ipynb#recent=true linki aracılığıyla erişilebilir. Bağlantıya tıklandığında karşınıza şu şekilde bir ekran gelmektedir.



Şekil 1. Google Colab ana ekranı

Gelen ekranın sağ alt köşesinde bulunan NEW NOTEBOOK tuşuna basılır ve yeni bri kod ekranı açılır. Şekil 2'de kodun yazılacağı sayfa gösterilmiştir.



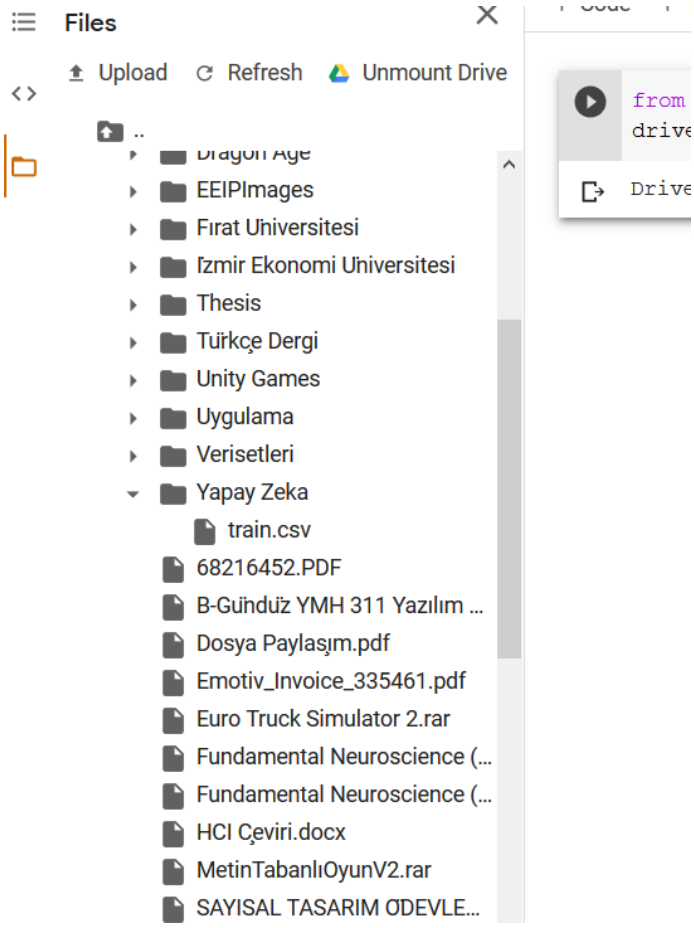
Şekil 2. Kodlama işleminin yapılacağı ekran

Spyder üzerinde yazılan kodlar benzer şekilde burada da yazılabilmekte ve gerekli sınıflandırma işlemleri burada da yapılabilmektedir. Google Colab ile analiz işlemi yapılırken en çok karşılaşılan problem sınıflandırma işlemi için kullanılacak olan dosyaların eklenmesidir. Normal lokal bir bilgisayarda çalışılırken, dosyanın yolunun sisteme gösterilmesi analiz işlemi için yeterliyken, Google Colab ile bu durum o kadar da kolay değildir. Bunun için öncelikle verilerinizin Google Drive'a aktarılması gerekmektedir. Google Colab, drivedaki verileri kullanarak dosya yollarını çekebilmektedir.

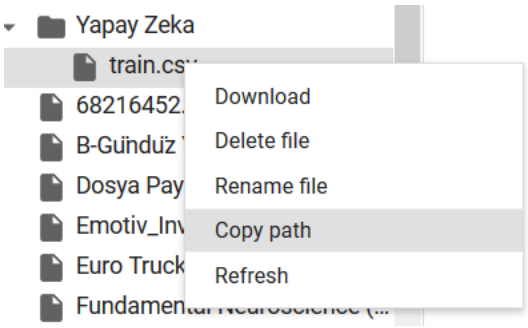
Aşağıdaki kod komutu ile Google Drive'a erişilir.

```
from google.colab import drive
drive.mount('/content/drive')
```

Kodu yazdıktan sonra sol tarafta drivedaki dosyalar görünecektir.



Ardından verimizin olduğu yolu kopyalayıp benzer işlemleri gerçekleştirebiliriz.



Önceki uygulamalarda gerçekleştirmiş olduğumuz YSA uygulamasını şimdi Google Colab üzerinde gerçekleştirebiliriz. Öncelikle gerekli kütüphaneler yüklenir ve verinin yolu belirlenir.

```
# kütüphanelerin yüklenmesi
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder

#verisetinin yüklenmesi
veri = pd.read_csv("/content/drive/My Drive/Yapay Zeka/train.csv")
```

Sınıf etiketleri, girdi ve çıktılar belirlenir.

```
#sınıf sayısını belirleme
label_encoder = LabelEncoder().fit(veri.price_range)
labels = label_encoder.transform(veri.price_range)
classes = list(label_encoder.classes_)

#Girdi ve çıktı verisinin hazırlanması
X = veri.drop(["price_range"],axis=1)
y = labels
```

Veriler standartlaştırılır ve eğitil ile test verileri oluşturulur.

```
#verilerin standartlaştırılması
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)

#eğitim ve test verilerinin hazırlanması
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2)
```

Çıktı değerleri kategorileştirilir ve YSA modeli oluşturulur.

```
#çıktı değerlerinin kategorileştirilmesi
from tensorflow.keras.utils import to_categorical
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

#YSA modelinin oluşturulması
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
model = Sequential()
model.add(Dense(16, input_dim=20, activation="relu"))
model.add(Dense(12, activation="relu"))
model.add(Dense(4, activation="softmax"))
model.summary()
```

Model derlenir ve eğitilir. Sonuçlar sistem üzerinden gösterilir.

```
#modelin derlenmesi
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

#modelin eğitilmesi
model.fit(X_train, y_train,validation_data = (X_test,y_test), epochs=100)

print(("Ortalama eğitim kaybı: ", np.mean(model.history.history["loss"])))
print(("Ortalama eğitim başarımı: ", np.mean(model.history.history["accuracy"])))
print(("Ortalama doğrulama kaybı: ", np.mean(model.history.history["val_loss"])))
print(("Ortalama doğrulama başarımı: ", np.mean(model.history.history["val_accuracy"])))
```

```
Epoch 73/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0372 - accuracy: 0.9956 - val_loss: 0.1959 - val_accuracy: 0.9175
Epoch 74/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0360 - accuracy: 0.9956 - val_loss: 0.1968 - val_accuracy: 0.9175
Epoch 75/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0357 - accuracy: 0.9956 - val_loss: 0.2010 - val_accuracy: 0.9175
Epoch 76/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0341 - accuracy: 0.9956 - val_loss: 0.2006 - val_accuracy: 0.9225
Epoch 77/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0330 - accuracy: 0.9981 - val_loss: 0.1985 - val_accuracy: 0.9150
Epoch 78/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0322 - accuracy: 0.9975 - val_loss: 0.1983 - val_accuracy: 0.9150
Epoch 79/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0316 - accuracy: 0.9969 - val_loss: 0.2041 - val_accuracy: 0.9150
Epoch 80/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0305 - accuracy: 0.9975 - val_loss: 0.2099 - val_accuracy: 0.9200
Epoch 81/100
```

Eğitim ve doğrulama başarımları grafik üzerinde gösterilerek analiz işlemi gerçekleştirilir.

```
#Eğitim ve doğrulama başarımlarının gösterilmesi
import matplotlib.pyplot as plt
plt.plot(model.history.history['accuracy'])
plt.plot(model.history.history['val_accuracy'])
plt.title('Model Başarımı')
plt.ylabel('Başarım')
plt.xlabel('Epok')
plt.legend(['Eğitim', 'Test'], loc='upper left')
plt.show()

#Eğitim ve test kayıpların gösterilmesi
plt.plot(model.history.history['loss'])
plt.plot(model.history.history['val_loss'])
plt.title('Model kaybı')
plt.ylabel('Kayıp')
plt.xlabel('Epok')
plt.legend(['Eğitim', 'Test'], loc='upper left')
plt.show()
```

Uygulama 8. Değerlendirme Soruları

1) Uygulama 4.4'te yapmış olduğunuz sınıflandırma işlemini Google Colab ile tekrar yapınız **(25p).**
2) Uygulama 5.4'te yapmış olduğunuz sınıflandırma işlemini Google Colab ile tekrar yapınız **(25p).**
3) Uygulama 6.4'te yapmış olduğunuz sınıflandırma işlemini Google Colab ile tekrar yapınız **(25p).**
4) Uygulama 7.2'te yapmış olduğunuz sınıflandırma işlemini Google Colab ile tekrar yapınız **(25p).**