```
# LAB 01 - Advanced Deep Learning
```

LAB 01 - Advanced Deep Learning

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

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

Show hidden output

New Section

```
#data = pd.read_csv('/content/Airline_Delay_Cause.csv')
data = pd.read_csv('/content/drive/MyDrive/Khemis_univ/2024-2025_1st/AdvancedDeepLearning_M2/Airline_Delay_Cause.csv')
```

```
data
```

Show hidden output

```
data.info()
```

Show hidden output

```
data = data.drop(['carrier','carrier_name','airport','airport_name'],axis=1)
data
```

Show hidden output

```
data.dropna(inplace=True)
```

```
data.info()
```

Show hidden output

```
data['weather_delay'].min(),data['weather_delay'].max()
```

Show hidden output

```
data['WDCase'] = data['weather_delay'].apply(lambda x : 1 if x >100 else 0)
```

```
data['WDCase'].value_counts()
```

Show hidden output

```
data
```

Show hidden output

```
X = data.drop(['WDCase'],axis = 1)
y =data['WDCase']
```

```

X
```

Show hidden output

```
y
```

Show hidden output

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=44, shuffle =True)

print('X_train shape is ' , X_train.shape)
print('X_test shape is ' , X_test.shape)
print('y_train shape is ' , y_train.shape)
print('y_test shape is ' , y_test.shape)
```

Show hidden output

```
import tensorflow as tf
import keras
```

```
KerasModel = keras.models.Sequential([
               # keras.layers.Input(shape=(17)),
            keras.layers.Dense(8,  activation = 'tanh'),
            # keras.layers.Dropout(0.1),
            keras.layers.Dense(128, activation = 'sigmoid'),
            # keras.layers.Dropout(0.3),
            keras.layers.Dense(64, activation = 'tanh'),
            keras.layers.Dense(32, activation = 'tanh'),
            keras.layers.Dropout(0.2),
            keras.layers.Dense(1, activation = 'sigmoid')
            ])
```

```
MyOptimizer = tf.keras.optimizers.AdamW(
    learning_rate=0.001,
    weight_decay=0.004,
    beta_1=0.9,
    beta_2=0.999,
    epsilon=1e-07,
    amsgrad=False,
    clipnorm=None,
    clipvalue=None,
    global_clipnorm=None,
    use_ema=False,
    ema_momentum=0.99,
    ema_overwrite_frequency=None,
    #jit_compile=True,
    name="AdamW")
```

```
KerasModel.compile(optimizer =MyOptimizer,loss='binary_crossentropy',metrics=['accuracy']) # matrix
```

```
history = KerasModel.fit(X_train,y_train,
                         validation_data=(X_test,y_test),
                         epochs=100,
                         batch_size=10000,
                         verbose=1,
                         callbacks=[tf.keras.callbacks.EarlyStopping(
                                          patience=10,
                                          monitor='val_accuracy',#"val_loss",
                                          restore_best_weights=True)])

```

Show hidden output

```
print(KerasModel.summary())
```

Show hidden output

```
KerasModel.save('KerasModel.model.keras')
```

```
NewKerasModel = keras.models.load_model('KerasModel.model.keras')

```

```
X_test
```

Show hidden output

```
y_pred = NewKerasModel.predict(X_test)
```

Show hidden output

```
print('Prediction Shape is {}'.format(y_pred.shape))
```

Show hidden output

```
print('Prediction items are {}'.format(y_pred[:5]))
```

Show hidden output

```
y_test[:5]
```

Show hidden output

```

ModelLoss, ModelAccuracy = NewKerasModel.evaluate(X_test, y_test)
print('Model Loss is {}'.format(ModelLoss))
print('Model Accuracy is {}'.format(ModelAccuracy ))

```

Show hidden output

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()

```

Show hidden output

```
# "Loss"
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()

```

Show hidden output

```

len(y_test), len(y_pred)

```

Show hidden output

```

y_pred

```

Show hidden output

```
y_pred = [np.round(i[0]) for i in y_pred]
y_pred
```

Show hidden output

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
CM = confusion_matrix(y_test, y_pred)
print('Confusion Matrix is : \n', CM)

```

Show hidden output

```

# drawing confusion matrix
sns.heatmap(CM, center = True,cmap='Blues_r')
plt.show()

```

Show hidden output

```

from sklearn.metrics import classification_report
ClassificationReport = classification_report(y_test,y_pred)
print('Classification Report is : \n', ClassificationReport )

```

Show hidden output

```
Start coding or generate with AI.
```