Boston

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

df = pd.read_csv('Boston.csv')
df.head()

df.info()

df.drop(columns= ['Unnamed: 15', 'Unnamed: 16', 'CAT. MEDV'], inplace = True)
df = df.dropna()
df.info()

corr_matrix = df.corr()
corr_matrix

corr_matrix['MEDV'].sort_values()

X = df.drop(columns = ['MEDV'])
y = df['MEDV']

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

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)

model = keras.Sequential([
    layers.InputLayer(shape=(X_train_scaled.shape[1],)),
    layers.Dense(64, activation = 'relu'),
    layers.Dense(32, activation = 'relu'),
    layers.Dense(1)
])

model.compile(optimizer = 'adam', loss = 'mse', metrics = ['mae'])

history = model.fit(X_train_scaled, y_train, epochs = 100, batch_size = 32, validation_data = (X_test_scaled, y_test), verbose = 1)

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.title('Training and Validation Loss')
plt.show()

y_pred = model.predict(X_test_scaled)

mse = mean_squared_error(y_test,y_pred)
mae = mean_absolute_error(y_test,y_pred)
r2 = r2_score(y_test,y_pred)

plt.scatter(y_test, y_pred, color='skyblue', label='Predictions')
plt.xlabel('True Values')
plt.ylabel('Predicted Values')
plt.title('True vs Predicted Values')
plt.plot(y_test, y_test, "r--", label='Regression Line')
plt.legend()
plt.grid(True)
plt.show()
```

IMDB

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import string

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.feature_extraction.text import TfidfVectorizer

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam

df = pd.read_csv('imdb.csv')
df.head()

df.info()

df.describe()

df.isnull().sum()

def clean_text(text):
    text = text.lower()
    text = re.sub(r'\[.*?\]', '', text)  # remove brackets
    text = re.sub(r'https?://\S+|www\.\S+', '', text)  # remove URLs
    text = re.sub(r'<.*?>+', '', text)  # remove HTML
    text = re.sub(r'[%s]' % re.escape(string.punctuation), '', text)  # remove punctuation
    text = re.sub(r'\n', '', text)
    text = re.sub(r'\w*\d\w*', '', text)  # remove numbers
    return text

df['cleaned_review'] = df['review'].apply(clean_text)

vectorizer = TfidfVectorizer(max_features=5000, stop_words='english')
X = vectorizer.fit_transform(df['cleaned_review']).toarray()
y = df['sentiment'].astype(int)

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

model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(X.shape[1],)))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])

history = model.fit(X_train, y_train, epochs=20, batch_size=64, validation_split=0.2)

y_pred = (model.predict(X_test) > 0.5).astype("int32")

print("Classification Report:\n", classification_report(y_test, y_pred))

print("Accuracy:", accuracy_score(y_test, y_pred))

conf_mat = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_mat, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

MNIST

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam

train_df = pd.read_csv('fashion-mnist_train.csv')
test_df = pd.read_csv('fashion-mnist_test.csv')

train_df.head()

train_df['label'].unique()

plt.figure(figsize=(12, 8))

for i in range(10):
    img = train_df[train_df['label'] == i].iloc[0, 1:].values.reshape(28, 28)
    plt.subplot(2, 5, i+1)
    plt.imshow(img, cmap='gray')
    plt.title(i)
    plt.axis('off')

plt.suptitle('Fashion MNIST Sample Images for Each Label')
plt.tight_layout()
plt.show()

label_names = {
    0: "T-shirt/top",
    1: "Trouser",
    2: "hoodie",
    3: "Dress",
    4: "Coat",
    5: "Sandal",
    6: "Shirt",
    7: "Sneaker",
    8: "Bag",
    9: "Shoe"
}

plt.figure(figsize=(12, 8))

for i in range(10):
    img = train_df[train_df['label'] == i].iloc[0, 1:].values.reshape(28, 28)
    plt.subplot(2, 5, i+1)
    plt.imshow(img, cmap='gray')
    plt.title(label_names[i])
    plt.axis('off')

plt.suptitle('Fashion MNIST Sample Images for Each Label')
plt.tight_layout()
plt.show()

X_train = train_df.drop('label', axis=1).values
y_train = train_df['label'].values
X_test = test_df.drop('label', axis=1).values
y_test = test_df['label'].values

X_train = X_train / 255.0
X_test = X_test / 255.0
X_train = X_train.reshape(-1, 28, 28, 1)
X_test = X_test.reshape(-1, 28, 28, 1)

y_train_cat = to_categorical(y_train, 10)
y_test_cat = to_categorical(y_test, 10)

model = Sequential()
model.add(Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(X_train, y_train_cat, epochs=10, validation_data=(X_test, y_test_cat), batch_size=128)

loss, accuracy = model.evaluate(X_test, y_test_cat)
print(f"Test Accuracy: {accuracy:.4f}")

y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
print(classification_report(y_test, y_pred_classes))

plt.figure(figsize=(8,6))
sns.heatmap(confusion_matrix(y_test, y_pred_classes), annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

Google

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout

dataset_train = pd.read_csv('Google_Stock_Price_Train.csv')

dataset_train.head()

training_set = dataset_train.iloc[:, 1: 2].values

training_set.shape

plt.figure(figsize=(18, 8))
plt.plot(dataset_train['Open'])
plt.title("Google Stock Open Prices")
plt.xlabel("Time (oldest -> latest)")
plt.ylabel("Stock Open Price")
plt.show()

sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(training_set)

X_train = []
y_train = []
for i in range(60, len(training_set_scaled)):
   X_train.append(training_set_scaled[i-60: i, 0])
   y_train.append(training_set_scaled[i, 0])

X_train, y_train = np.array(X_train), np.array(y_train)

X_train.shape

y_train.shape

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

X_train.shape

regressor = Sequential()
regressor.add(LSTM(units = 50, return_sequences = True, input_shape =
(X_train.shape[1], 1)))
regressor.add(Dropout(rate = 0.2))
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(rate = 0.2))
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(rate = 0.2))
regressor.add(LSTM(units = 50, return_sequences = False))
regressor.add(Dropout(rate = 0.2))
regressor.add(Dense(units = 1))

regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

regressor.fit(x = X_train, y = y_train, batch_size = 32, epochs = 50)

dataset_test = pd.read_csv('Google_Stock_Price_Test.csv')

dataset_test.head()

real_stock_price = dataset_test.iloc[:, 1: 2].values
real_stock_price.shape

dataset_total = pd.concat((dataset_train['Open'], dataset_test['Open']),
axis = 0)
inputs = dataset_total[len(dataset_total) - len(dataset_test) - 60:].values
inputs = inputs.reshape(-1, 1)
inputs = sc.transform(inputs)

len(inputs)

X_test = []
for i in range(60, len(inputs)):
   X_test.append(inputs[i-60:i, 0])
X_test = np.array(X_test)
#add dimension of indicator
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

X_test.shape

predicted_stock_price = regressor.predict(X_test)

predicted_stock_price = sc.inverse_transform(predicted_stock_price)

plt.plot(real_stock_price, color = 'red', label = 'Real price')
plt.plot(predicted_stock_price, color = 'blue', label = 'Predicted price')
plt.title('Google price prediction')
plt.xlabel('Time')
plt.ylabel('Price')
plt.legend()
plt.show()
```

OCR letter

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler , LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/letter-recognition/letter-recognition.data"
columns = ['letter', 'x-box', 'y-box', 'width', 'height', 'onpix', 'x-bar', 'y-bar', 'x2bar',
        'y2bar', 'xybar', 'x2ybr', 'xy2br', 'x-ege', 'xegvy', 'y-ege', 'yegvx']

df = pd.read_csv(url, names=columns)
df

x=df.drop('letter',axis=1)

y=df['letter']

#Appy the standard scaler on x
scaler=StandardScaler()
x_scaled=scaler.fit_transform(x)

#Apply Label encoder and one hot encoding to y

le=LabelEncoder()
y_encoded=le.fit_transform(y)
y_onehot=to_categorical(y_encoded)

x_train,x_test,y_train,y_test=train_test_split(x_scaled,y_onehot,test_size=0.2)

model=Sequential()
model.add(Dense(128,activation='relu',input_shape=(x_train.shape[1],))
)
model.add(Dense(64,activation='relu'))
model.add(Dense(26,activation='softmax'))

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

model.fit(x_train,y_train,validation_split=0.2,epochs=30,batch_size=64)

loss,accuracy=model.evaluate(x_test,y_test)

from sklearn.metrics import confusion_matrix,classification_report

y_pred=model.predict(x_test)

predicted_labels=le.inverse_transform(np.argmax(y_pred,axis=1))
actual_labels=le.inverse_transform(np.argmax(y_test,axis=1))

cm=confusion_matrix(actual_labels,predicted_labels,labels=le.classes_)
cm

import seaborn as sns

sns.heatmap(cm,annot=True)

print(classification_report(actual_labels,predicted_labels))
```

Human face recognition

```
import cv2

face_cap = cv2.CascadeClassifier("Unconfirmed 526091.crdownload")
video = cv2.VideoCapture(0)

while True:
    ret, frame = video.read()
    if not ret:
        break

    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = face_cap.detectMultiScale(
        gray_frame,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(30, 30),
        flags=cv2.CASCADE_SCALE_IMAGE
    )

    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    cv2.imshow("Real-Time Face Detection", frame)

    if cv2.waitKey(1) & 0xFF == ord('t'):
        break

video.release()
cv2.destroyAllWindows()
```