Spam Email Classifier Project

Objective:

Build a machine learning model that classifies emails as spam or not spam.

Dataset:

Used the 'SMS Spam Collection' dataset from UCI repository.

Libraries Used:

- pandas
- sklearn
- matplotlib
- seaborn

Steps:

1. Load and clean data
2. Feature extraction using TF-IDF Vectorizer
3. Train model using Naive Bayes
4. Evaluate performance

Code Summary:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
```

```
from sklearn.metrics import accuracy_score, classification_report

# Load data
data = pd.read_csv('spam.csv', encoding='latin-1')[['v1', 'v2']]
data.columns = ['label', 'text']
data['label'] = data['label'].map({'ham': 0, 'spam': 1})

# Split data
X_train, X_test, y_train, y_test = train_test_split(data['text'], data['label'], test_size=0.2)

# Vectorization
tfidf = TfidfVectorizer()
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)

# Train model
model = MultinomialNB()
model.fit(X_train_tfidf, y_train)

# Evaluate
predictions = model.predict(X_test_tfidf)
print(accuracy_score(y_test, predictions))
print(classification_report(y_test, predictions))
```

Image Classifier for Fruits Project

Objective:

Build a CNN model to classify fruit images.

Dataset:

Used a small fruit images dataset with 3 categories: apple, banana, orange.

Libraries Used:

- tensorflow
- keras
- matplotlib

Steps:

1. Load and preprocess image data
2. Build CNN model
3. Train and evaluate model

Code Summary:

```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

# Image preprocessing
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
```

```
train_data = train_datagen.flow_from_directory('fruits', target_size=(100, 100), class_mode='categorical', subset='training')
val_data = train_datagen.flow_from_directory('fruits', target_size=(100, 100), class_mode='categorical', subset='validation')

# Build model
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(100,100,3)),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(3, activation='softmax')
])

# Compile and train
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(train_data, validation_data=val_data, epochs=10)
```