Date		Session No	6

Topic : BI-Unit 3 - ML Basics

Foundations of Machine Learning: Neural Networks, SVM, and Sentiment Analysis

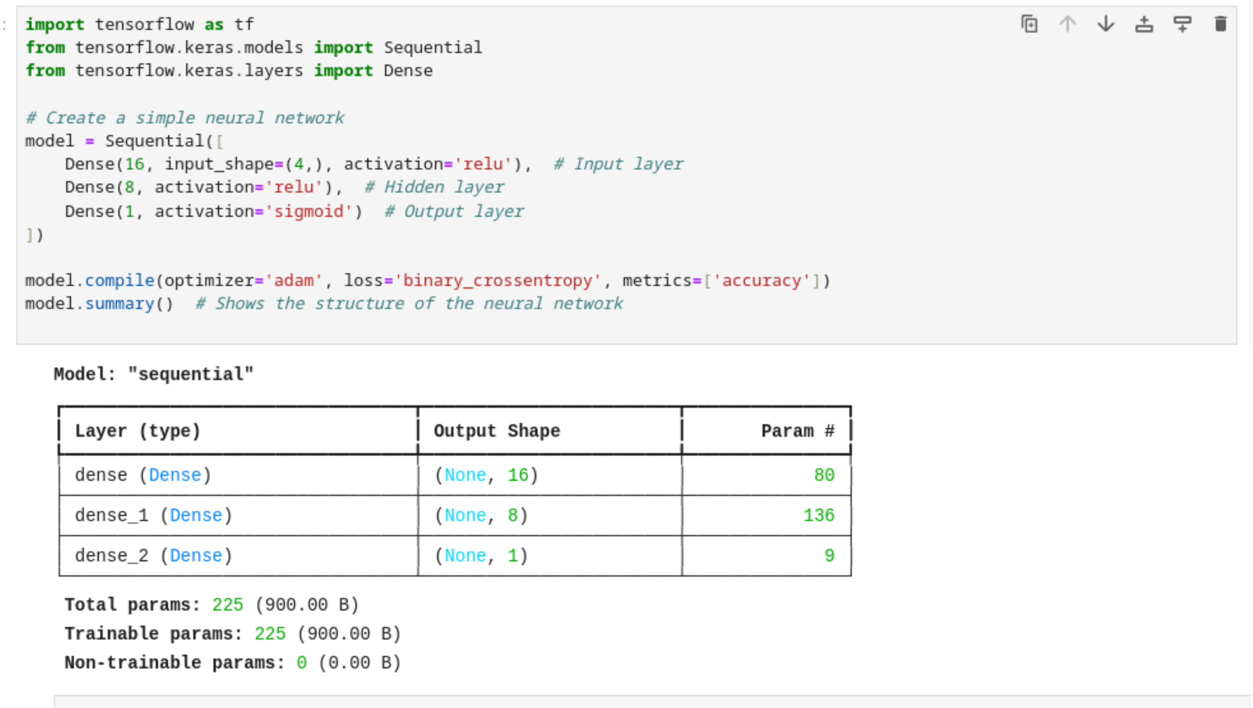
1. Basic Concepts of Neural Networks

A **Neural Network (NN)** is inspired by the structure of the human brain. It consists of layers of nodes (neurons) that process and learn from data to make predictions or decisions.

- **Input Layer**: Receives the data.
- **Hidden Layer(s)**: Where computations take place.
- **Output Layer**: Provides the result.

Each neuron connects to other neurons with **weights**, and the network adjusts these weights during training to improve accuracy.

Python Example (Using TensorFlow/Keras):



```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Create a simple neural network
model = Sequential([
    Dense(16, input_shape=(4,), activation='relu'),  # Input layer
    Dense(8, activation='relu'),  # Hidden layer
    Dense(1, activation='sigmoid')  # Output layer
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.summary()  # Shows the structure of the neural network
```

```
Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                     ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ dense (Dense)                    │ (None, 16)             │            80 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                  │ (None, 8)              │           136 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_2 (Dense)                  │ (None, 1)              │             9 │
└──────────────────────────────────┴────────────────────────┴───────────────┘

 Total params: 225 (900.00 B)
 Trainable params: 225 (900.00 B)
 Non-trainable params: 0 (0.00 B)
```

2. Developing Neural Network-Based Systems

To develop a neural network-based system:

1. **Data Preparation**: Gather and preprocess data (normalize, clean, etc.).
2. **Model Design**: Choose the network architecture (number of layers, neurons, activation functions).
3. **Training**: Feed data into the network, and it adjusts weights based on the error.
4. **Testing and Validation**: Test with unseen data to evaluate performance.

---

3. Illuminating the Black Box of ANN with Sensitivity

Artificial Neural Networks (ANNs) are often considered a "black box" because it's difficult to interpret how they make decisions. **Sensitivity Analysis** helps understand the influence of input features on the output.

Example:

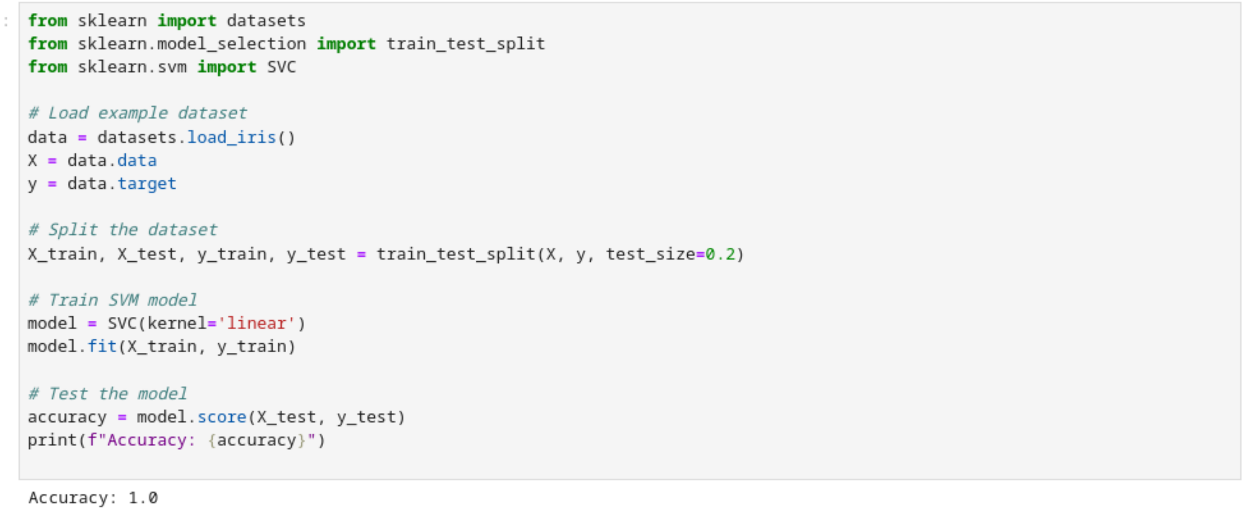
- If a model predicts house prices, you can test how sensitive the output is to changes in square footage, number of bedrooms, etc.

4. Support Vector Machines (SVM)

Support Vector Machines are a type of machine learning algorithm used for classification tasks. SVM finds the optimal boundary (hyperplane) that separates different classes of data with the maximum margin.

- SVM works well for high-dimensional data and can be used for both linear and non-linear classification.

Python Example (Using scikit-learn):



```python
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

# Load example dataset
data = datasets.load_iris()
X = data.data
y = data.target

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Train SVM model
model = SVC(kernel='linear')
model.fit(X_train, y_train)

# Test the model
accuracy = model.score(X_test, y_test)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 1.0
```

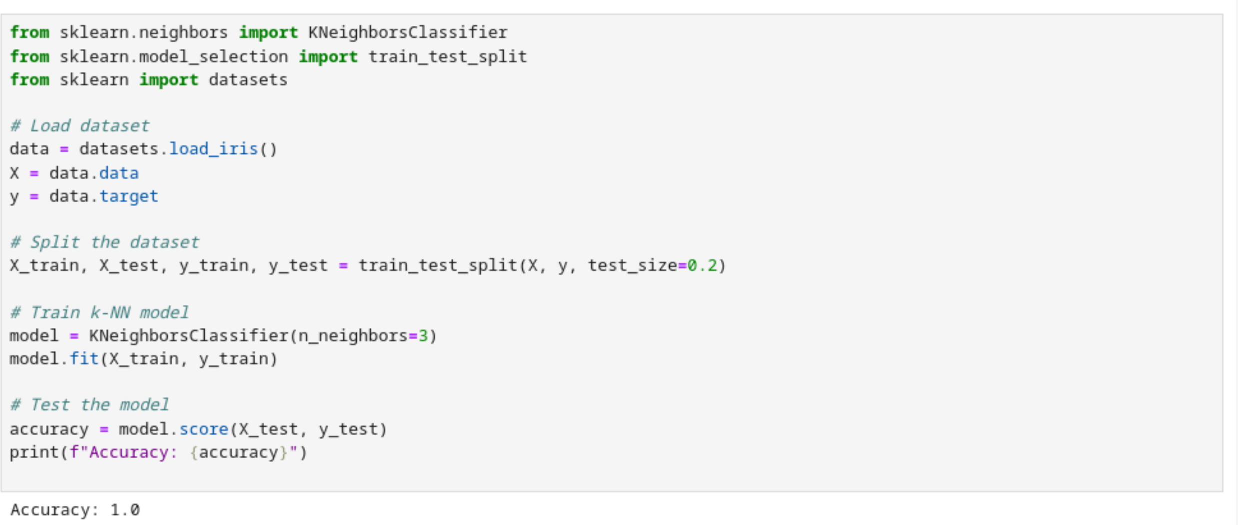
5. A Process-Based Approach to the Use of SVM

1. **Data Collection**: Collect data and label it according to the problem (classification).
2. **Feature Extraction**: Identify important features of the data.
3. **Model Training**: Train the SVM model using the training dataset.
4. **Model Testing**: Test the trained SVM on unseen data and measure accuracy.
5. **Tuning**: Adjust parameters like kernel type (linear, polynomial, radial) for better performance.

6. Nearest Neighbour Method for Prediction

The **k-Nearest Neighbour (k-NN)** algorithm is a simple, non-parametric method used for classification or regression. It predicts the output by finding the 'k' closest data points (neighbors) and using their majority class or average.

Python Example (Using scikit-learn):



```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import datasets

# Load dataset
data = datasets.load_iris()
X = data.data
y = data.target

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Train k-NN model
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)

# Test the model
accuracy = model.score(X_test, y_test)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 1.0
```

7. Sentiment Analysis Overview

Sentiment Analysis is a technique used to determine the emotional tone behind a body of text, such as identifying whether a sentence is positive, negative, or neutral.

It is widely used in social media monitoring, customer feedback analysis, and product reviews.

---

8. Sentiment Analysis Applications

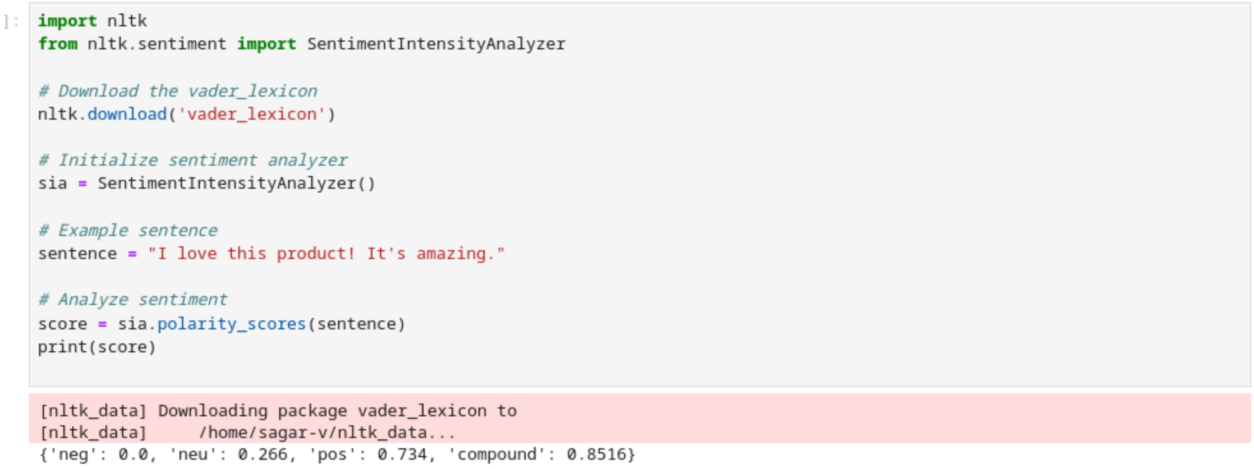
- **Social Media Monitoring**: Analyze tweets, posts, and comments to understand public sentiment on a topic.
- **Product Reviews**: Determine customer satisfaction based on product feedback.
- **Customer Service**: Analyze support tickets to detect frustration or satisfaction.

---

9. Sentiment Analysis Process

1. **Data Collection**: Gather text data from sources like social media, reviews, or support tickets.
2. **Preprocessing**: Clean the text (remove stopwords, punctuation, etc.).
3. **Tokenization**: Split the text into individual words (tokens).
4. **Model Training**: Train a machine learning or deep learning model to classify the sentiment.
5. **Analysis**: Predict sentiment and interpret the results.

Python Example (Using NLTK for Sentiment Analysis):



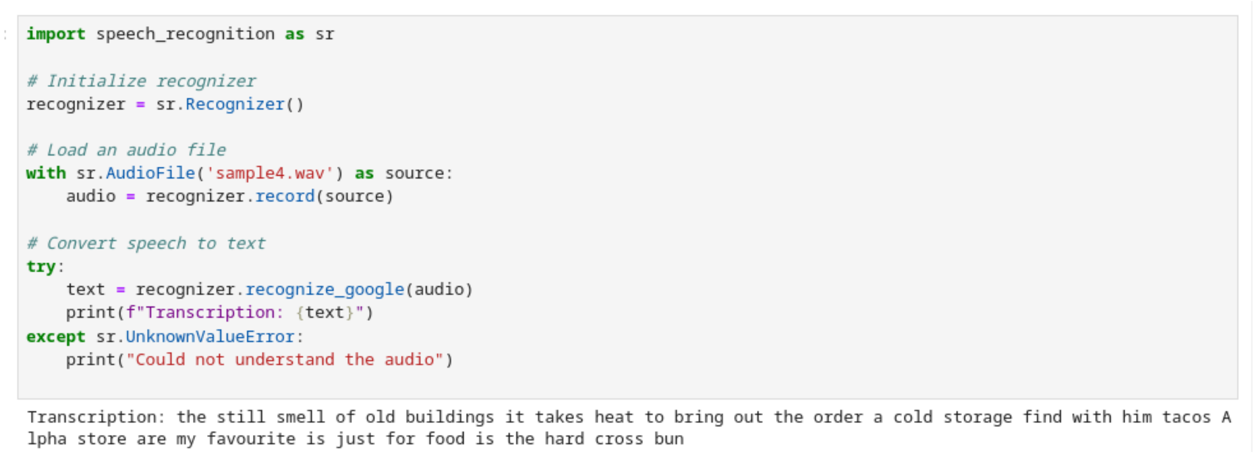
```
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer

# Download the vader_lexicon
nltk.download('vader_lexicon')

# Initialize sentiment analyzer
sia = SentimentIntensityAnalyzer()

# Example sentence
sentence = "I love this product! It's amazing."

# Analyze sentiment
score = sia.polarity_scores(sentence)
print(score)
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /home/sagar-v/nltk_data...
{'neg': 0.0, 'neu': 0.266, 'pos': 0.734, 'compound': 0.8516}
```

10. Speech Analytics

Speech Analytics is the process of analyzing recorded audio conversations to extract useful information such as emotion, sentiment, and intent. It's used in call centers, customer service, and speech recognition systems.

Python Example (Using `SpeechRecognition` library for basic speech-to-text):



```
import speech_recognition as sr

# Initialize recognizer
recognizer = sr.Recognizer()

# Load an audio file
with sr.AudioFile('sample4.wav') as source:
    audio = recognizer.record(source)

# Convert speech to text
try:
    text = recognizer.recognize_google(audio)
    print(f"Transcription: {text}")
except sr.UnknownValueError:
    print("Could not understand the audio")
```

```
Transcription: the still smell of old buildings it takes heat to bring out the order a cold storage find with him tacos A
lpha store are my favourite is just for food is the hard cross bun
```

END