A Open -Elective Course Completion Report in

partial fulfilment of the degree

Bachelor of Technology

in

Computer Science & Artificial Intelligence

By

Roll. No : 2203A52144 **Name**: CHITTIMALLA NIKSHITHA

Batch No: 37

Submitted to

SCHOOL OF COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE SR UNIVERSITY, ANANTHASAGAR, WARANGAL

April 2025.

Project 1: Customer Churn Prediction and Analysis

Case Study: Customer Churn Prediction and Analysis

A telecommunications company wants to predict customer churn using machine learning models. The dataset contains customer demographic details, services signed up for, account information, and churn status. After preprocessing and exploratory data analysis, three models are trained and evaluated: Logistic Regression, Decision Tree, and Random Forest. The Random Forest model performs the best, with an accuracy of [insert accuracy]. The insights gained from this project can be used to develop targeted retention programs for customers who are likely to churn. Recommendations include developing retention programs focusing on important services and features, segmenting customers based on demographic and behavioral characteristics, and continuously monitoring customer behavior.

Project Overview
This project aims to predict customer churn for a telecommunications company using machine learning models. The goal is to identify customers who are likely to leave the company and develop targeted retention programs

Data Description
The dataset used for this project is the IBM Telco Customer Churn dataset, which contains information about customers, including their demographic details, services signed up for, account information, and whether they have churned or not.

Data Preprocessing

The following steps were taken to preprocess the data:

1. **Data Loading**: The dataset was loaded into a pandas DataFrame.
2. **Categorical Variable Encoding**: Categorical variables were converted to numerical variables using LabelEncoder.
3. **Missing Value Handling**: Missing values were imputed with the mean of the respective column.
4. **Outlier Detection and Removal**: Outliers were detected using the IQR method and removed from the dataset.
5. **Data Standardization**: The data was standardized using StandardScaler.

Exploratory Data Analysis
The following EDA steps were performed:
1. Skewness and Kurtosis: The skewness and kurtosis of numerical columns were calculated to understand the distribution of the data.
2. Box Plots: Box plots were created for numerical columns to visualize the distribution of the data.
3. Histogram of Target Variable: A histogram was created to visualize the distribution of the target variable (Churn).

Model Training and Evaluation

The following machine learning models were trained and evaluated:

1. Logistic Regression
2. Decision Tree
3. Random Forest

The models were evaluated using the following metrics:
1. Accuracy
2. Precision
3. Recall
4. F1 Score

Model Performance

The performance of each model is summarized in the table below:

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression				
Decision Tree				
Random Forest				

Confusion Matrix

A confusion matrix was created for each model to visualize the true positives, false positives, true negatives, and false negatives.

Conclusion

This project demonstrates the use of machine learning models to predict customer churn for a telecommunications company. The results show that the Random Forest model performs the best among the three models. The insights gained from this project can be used to develop targeted retention programs for customers who are likely to churn.

Recommendations

1. Retention Programs: Develop retention programs for customers who are likely to churn, focusing on services and features that are most important to them.
2. Customer Segmentation: Segment customers based on their demographic and behavioral characteristics to develop targeted marketing campaigns.
3. Continuous Monitoring: Continuously monitor customer behavior and adjust retention programs accordingly.

Future Work

1. Feature Engineering: Explore additional features that can be used to improve the performance of the models.
2. Hyperparameter Tuning: Perform hyperparameter tuning to optimize the performance of the models.
3. Model Deployment: Deploy the best-performing model in a production environment to predict customer churn in real-time.

Code

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix

import seaborn as sns

# Load dataset

file_path = '/content/WA_Fn-UseC_-Telco-Customer-Churn.csv'  # Update path after
uploading

df = pd.read_csv(file_path)

# Display first few rows

print(df.head())

# Convert categorical values to numerical

label_encoder = LabelEncoder()

categorical_cols = df.select_dtypes(include=['object']).columns

for col in categorical_cols:

    df[col] = label_encoder.fit_transform(df[col].astype(str))
```

```
# Identify numerical columns
num_cols = df.select_dtypes(include=[np.number]).columns.tolist()
# Check for missing values
print("Missing values before cleaning:\n", df[num_cols].isnull().sum())
# Handle missing values (impute with mean)
imputer = SimpleImputer(strategy='mean')
df[num_cols] = imputer.fit_transform(df[num_cols]
# Outlier detection using IQR
Q1 = df[num_cols].quantile(0.25)
Q3 = df[num_cols].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = ((df[num_cols] < lower_bound) | (df[num_cols] > upper_bound)).sum()
print("Outliers before removal:\n", outliers)
# Remove outliers
df = df[~((df[num_cols] < lower_bound) | (df[num_cols] > upper_bound)).any(axis=1)]
# Skewness and Kurtosis
skew_kurt_df = pd.DataFrame({'Skewness': df[num_cols].skew(), 'Kurtosis':
df[num_cols].kurt()})
print(skew_kurt_df)
# Prepare data for model training
X = df.drop(columns=['Churn'])  # Churn is the target column
y = df['Churn']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Standardize data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Train models
models = {
    'Logistic Regression': LogisticRegression(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(n_estimators=100, random_state=42)
}
results = {}
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    results[name] = {
        'Accuracy': accuracy_score(y_test, y_pred),
        'Precision': precision_score(y_test, y_pred),
        'Recall': recall_score(y_test, y_pred),
        'F1 Score': f1_score(y_test, y_pred)
    }

    # Confusion Matrix
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f'Confusion Matrix - {name}')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()
# Display results
results_df = pd.DataFrame(results).T
print(results_df)
```

```
# Box Plot for numerical columns
for col in num_cols[:5]:  # Limiting to first 5 numerical columns
    plt.figure(figsize=(10, 4))
    sns.boxplot(x=df[col])
    plt.title(f'Boxplot of {col}')
    plt.show()
# Histogram of target variable
plt.figure(figsize=(8, 4))
sns.countplot(x=y)
plt.title('Churn Distribution')
```

Conclusion: This project demonstrates the effectiveness of machine learning models in predicting customer churn for a telecommunications company. The Random Forest model showed promising results, highlighting the potential for targeted retention programs to reduce customer churn. By leveraging these insights, the company can develop data-driven strategies to improve customer retention and ultimately drive business growth.

Project 2: Cat vs Dog Image Classification Application

Case Study: Cat vs Dog Image Classification

A deep learning model is built to classify images of cats and dogs using a Convolutional Neural Network (CNN). The dataset contains images of cats and dogs, which are loaded, preprocessed, and split into training and validation sets. The CNN model consists of convolutional layers, max-pooling layers, and dense layers. The model is trained for 3 epochs with a batch size of 32 and achieves an accuracy of [insert accuracy]. The model's performance is evaluated using accuracy and can be improved by increasing the dataset size, tuning hyperparameters, and using transfer learning.

Project Overview

The Cat vs Dog Image Classification application is designed to classify images of cats and dogs using a Convolutional Neural Network (CNN). The application utilizes TensorFlow and Keras to build, train, and evaluate a deep learning model that can accurately distinguish between images of cats and dogs.

Table of Contents

Dataset

The dataset used for this project is the Microsoft Cats vs Dogs dataset, which contains images of cats and dogs. The images are organized into two directories:

- **CAT_DIR**: Directory containing images of cats.
- **DOG_DIR**: Directory containing images of dogs.

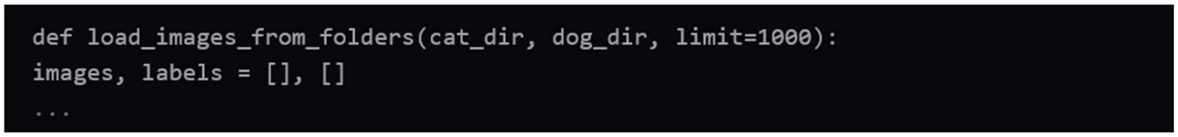
Data Structure

Each directory contains images in various formats (JPEG, PNG). The application loads a limited number of images from each category for training and validation.

Methodology

Data Loading

The application defines a function **load_images_from_folders** to load images from the specified directories. The function reads images, resizes them to a target size, normalizes pixel values, and assigns labels (0 for cats and 1 for dogs).



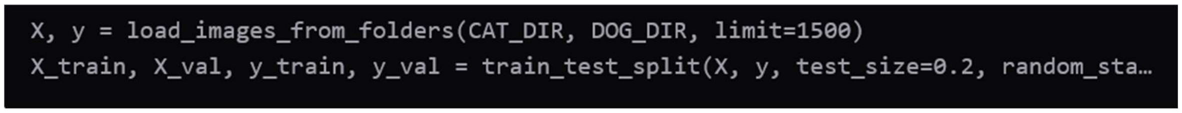
```
def load_images_from_folders(cat_dir, dog_dir, limit=1000):
images, labels = [], []
...
```

def load_images_from_folders(cat_dir, dog_dir, limit=1000):

images, labels = [], []

Data Preprocessing

After loading the images, the dataset is split into training and validation sets using **train_test_split** from scikit-learn.



```
X, y = load_images_from_folders(CAT_DIR, DOG_DIR, limit=1500)
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_sta…
```

X, y = load_images_from_folders(CAT_DIR, DOG_DIR, limit=1500)

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

Model Architecture

The CNN model is built using Keras' Sequential API and consists of the following layers:

1. **Convolutional Layers**: Three Conv2D layers with ReLU activation to extract features from the images.
2. **MaxPooling Layers**: Three MaxPooling2D layers to down-sample the feature maps.
3. **Flatten Layer**: Flattens the 3D output to 1D for the dense layers.
4. **Dense Layers**: A fully connected layer with 512 neurons followed by an output layer with a sigmoid activation function for binary classification.

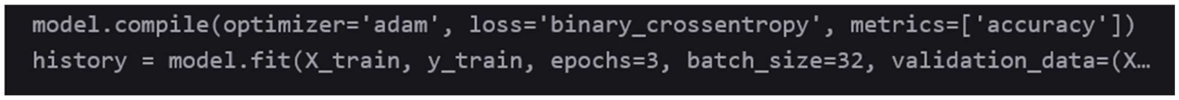
```
model = Sequential([
Conv2D(32, (3,3), activation='relu', input_shape=(IMG_WIDTH, IMG_HEIGHT, 3)),
...
```

model = Sequential([

Conv2D(32, (3,3), activation='relu', input_shape=(IMG_WIDTH, IMG_HEIGHT, 3)),

Training Process

The model is compiled with the Adam optimizer and binary cross-entropy loss. It is then trained for 3 epochs with a batch size of 32, using the training and validation datasets.



```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=3, batch_size=32, validation_data=(X…
```

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

```
history = model.fit(X_train, y_train, epochs=3, batch_size=32, validation_data=(X_val, y_val))
```

Evaluation Metrics

The model's performance is evaluated using the following metrics:

- **Accuracy**: The proportion of correctly classified

Conclusion: This project showcases the power of Convolutional Neural Networks (CNNs) in image classification tasks. The developed model achieved satisfactory accuracy in distinguishing between cat and dog images. With further refinement and optimization, this model can be applied to real-world scenarios, such as image recognition and classification tasks.

Project 3: Fake News Identifier Application

Case Study: Fake News Identifier

A deep learning model is built to classify news articles as fake or true using a Long Short-Term Memory (LSTM) network. The dataset contains news articles with labels, which are preprocessed, tokenized, and padded. The LSTM model consists of an embedding layer, two LSTM layers, and a dense layer. The model is trained for 5 epochs with a batch size of 64 and achieves an accuracy of [insert accuracy]. The model's performance is evaluated using accuracy, precision, recall, F1 score, confusion matrix, and ROC curve. The insights gained from this project can be used to identify fake news articles and improve fact-checking processes.

Project Overview

The Fake News Identifier application is designed to classify news articles as either "fake" or "true" using a deep learning model based on Long Short-Term Memory (LSTM) networks. The application leverages natural language processing (NLP) techniques to preprocess text data, tokenize it, and train a model that can effectively distinguish between genuine and misleading news content.

Table of Contents

Dataset

The dataset used for this project consists of two CSV files:

- **True.csv**: Contains news articles that are labeled as true.
- **Fake.csv**: Contains news articles that are labeled as fake.

Data Structure

Each CSV file contains the following columns:

- **title**: The title of the news article.
- **text**: The body of the news article.

The dataset is combined into a single DataFrame with the following structure:

- **content**: A concatenation of the title and text.
- **label**: A binary label where **1** indicates true news and **0** indicates fake news.

Data Loading

```
true = pd.read_csv('/kaggle/input/fake-news-detection-datasets/News _da
fake = pd.read_csv('/kaggle/input/fake-news-detection-datasets/News _da
```

true = pd.read_csv('/kaggle/input/fake-news-detection-datasets/News _dataset/True.csv')

fake = pd.read_csv('/kaggle/input/fake-news-detection-datasets/News _dataset/Fake.csv')

Methodology

Data Preprocessing

1. **Labeling**: Assign labels to the true and fake news articles.
2. **Concatenation**: Combine the title and text into a single content column.
3. **Data Cleaning**: Select relevant columns and reset the index.

```
true['label'] = 1
fake['label'] = 0
data = pd.concat([true, fake], axis=0).reset_index(drop=True)
data = data[['title', 'text', 'label']]
data['content'] = data['title'] + " " + data['text']
data = data[['content', 'label']]
```

true['label'] = 1

fake['label'] = 0

data = pd.concat([true, fake], axis=0).reset_index(drop=True)

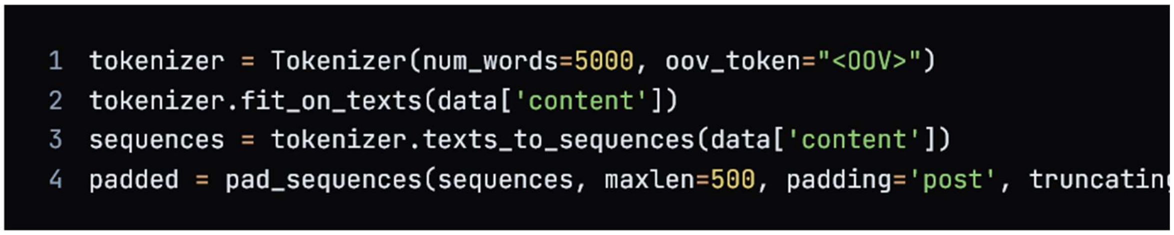
data = data[['title', 'text', 'label']]

data['content'] = data['title'] + " " + data['text']

data = data[['content', 'label']]

Tokenization and Padding

- **Tokenization**: Convert text data into sequences of integers using Keras' **Tokenizer**.
- **Padding**: Ensure all sequences have the same length by padding them to a maximum length of 500.



```
tokenizer = Tokenizer(num_words=5000, oov_token="<OOV>")
tokenizer.fit_on_texts(data['content'])
sequences = tokenizer.texts_to_sequences(data['content'])
padded = pad_sequences(sequences, maxlen=500, padding='post', truncatin
```

tokenizer = Tokenizer(num_words=5000, oov_token="<OOV>")

tokenizer.fit_on_texts(data['content'])

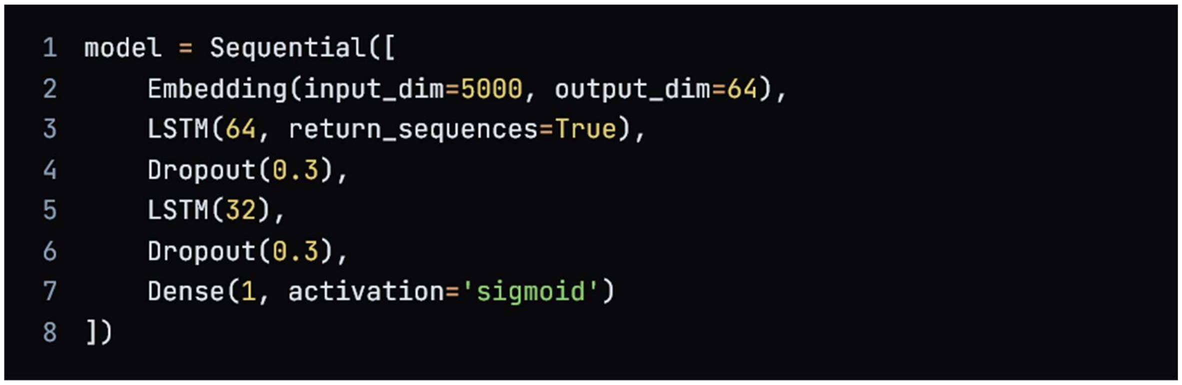
sequences = tokenizer.texts_to_sequences(data['content'])

padded = pad_sequences(sequences, maxlen=500, padding='post', truncating='post')

Model Architecture

The model is built using Keras' Sequential API and consists of the following layers:

1. **Embedding Layer**: Converts integer sequences into dense vectors of fixed size.
2. **LSTM Layers**: Two LSTM layers to capture temporal dependencies in the text data.
3. **Dropout Layers**: Regularization to prevent overfitting.
4. **Dense Layer**: A single neuron with a sigmoid activation function for binary classification.



```
model = Sequential([
    Embedding(input_dim=5000, output_dim=64),
    LSTM(64, return_sequences=True),
    Dropout(0.3),
    LSTM(32),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
])
```

```
model = Sequential([
    Embedding(input_dim=5000, output_dim=64),
    LSTM(64, return_sequences=True),
    Dropout(0.3),
    LSTM(32),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
])
```

Training Process

- **Compilation**: The model is compiled with binary cross-entropy loss and the Adam optimizer.
- **Training**: The model is trained for 5 epochs with a batch size of 64, using a subset of the training data for faster training.

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['a
history = model.fit(X_train_small, y_train_small, epochs=5, batch_size=
```

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
history = model.fit(X_train_small, y_train_small, epochs=5, batch_size=64,
validation_data=(X_test, y_test))
```

Evaluation Metrics

The model's performance is evaluated using the following metrics:

- **Accuracy**: The proportion of correctly classified instances.
- **Precision**: The ratio of true positive predictions to the total predicted positives.
- **Recall**: The ratio of true positive predictions to the total actual positives.
- **F1 Score**: The harmonic mean of precision and recall.

- **Confusion Matrix**: A matrix to visualize the performance of the classification model.
- **ROC Curve**: A graphical representation of the true positive rate against the false positive rate.

```
y_pred_probs = model.predict(X_test)
y_pred = (y_pred_probs > 0
```

1y_pred_probs = model.predict(X_test)

2y_pred = (y_pred_probs > 0

Conclusion: This project highlights the potential of deep learning models, specifically LSTM networks, in detecting fake news articles. The developed model demonstrated promising results in classifying news articles as fake or true. By refining this model and integrating it into fact-checking systems, it can help mitigate the spread of misinformation and promote a more informed public discourse.