Nilai UNIVERSITY

Faculty of Engineering, Science and Technology

School of Computing

Assignment Cover Sheet

Course Code: ________________________ Course Title: ____

Assignment Title: ____ Due Date: _______________

Date Submitted: _____________________ Lecturer Name: _________________________________

To be completed if this is an individual assignment

I declare that this assignment is my individual work. I have not worked collaboratively nor have I copied from any other student's work or from any other source except where due acknowledgement is made explicitly in the text, nor has any part been written for me by another person.

Student name: ________________________________ Student ID: _____________________________

Signature: __________________________________

To be completed if this is a group assignment

We declare that this is a group assignment and that no part of this submission has been copied from any other student's work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part been written for us by another person.

Student ID	Student Name	Signature

Lecturer's comments: ____

____

Total Marks: _________________________ Lecturer's Signature: _______________________

Feedback to Student:

I/We acknowledged receiving feedback from the lecturer on this assignment.

Student's Signature: ______________ _______________ _______________ _____________

Extension certification:

This assignment has been given an extension and is now due on ______________________

Lecturer's Signature: _____________________

Table of Content

Coverpage **0**

Table of Content **1**

1.0 Introduction **2**

2.0 Dataset Selection . . . . **3 - 6**

3.0 Model Selection & Evaluation . . . . **7 - 13**

4.0 Comparison & Justification. . . . . **14 - 18**

5.0 Conclusion **19**

1.0 Introduction

This assignment focuses on analyzing and communicating the evaluation of machine learning models for predicting customer churn in the telecommunications industry. Using the Telco Customer Churn Dataset, two models—Decision Tree and Neural Network—are implemented to predict churn based on customer attributes, service usage, and account information. The models are assessed using performance metrics such as accuracy, precision, recall, and F1-score, followed by a comparison to identify the best-performing model. The findings are then presented in a clear report, demonstrating the ability to evaluate models and apply data-driven decision-making in business contexts.

2.0 Dataset Selection

2.1 Data Set Overview

The Telco Customer Churn Dataset, sourced from Kaggle and based on IBM Sample Data Sets, focuses on customer retention in the telecommunications industry. It contains 7,043 rows and 21 columns, representing customer attributes like demographics (e.g., gender, senior citizen status), service subscriptions (e.g., phone, internet, streaming), and billing details (e.g., contract type, payment method). The goal is to predict customer churn, indicated by a binary variable (Yes/No) in the Churn column. This dataset is ideal for training machine learning models to identify at-risk customers and help businesses improve retention strategies.

2.2 Features of the Dataset

The Telco Customer Churn Dataset contains 21 features, which can be categorized into independent variables (input features) and the dependent variable (target feature). These features provide insights into customer demographics, service usage, and billing details, all of which influence customer churn.

Target Variable (Dependent Variable)

- Churn: Indicates whether a customer has left the company (Yes) or remained (No). This is the primary variable that the machine learning models aim to predict.

Independent Variables (Input Features)

- The independent variables can be grouped into three main categories:
    a. Demographic Features (Customer personal details)
        - gender: Specifies whether the customer is male or female.
        - SeniorCitizen: A binary variable indicating if the customer is a senior citizen (`1` for Yes, `0` for No).
        - Partner: Specifies whether the customer has a partner (`Yes` or `No`).
        - Dependents: Indicates whether the customer has dependents (`Yes` or `No`).

    b. Service-Related Features (Subscription details)
        - PhoneService: Indicates whether the customer has a phone service (`Yes` or `No`).
        - MultipleLines: Specifies if the customer has multiple lines (`Yes`, `No`, or `No phone service`).
        - InternetService: Type of internet service the customer has (`DSL`, `Fiber optic`, or `No`).
        - OnlineSecurity: Specifies if the customer has subscribed to an online security service (`Yes`, `No`, or `No internet service`).
        - OnlineBackup: Indicates whether the customer has an online backup service (`Yes`, `No`, or `No internet service`).
        - DeviceProtection: Shows whether the customer has device protection (`Yes`, `No`, or `No internet service`).
        - TechSupport: Indicates if the customer has technical support services (`Yes`, `No`, or `No internet service`).
        - StreamingTV: Specifies whether the customer has a streaming TV service (`Yes`, `No`, or `No internet service`).
        - StreamingMovies: Indicates if the customer has a streaming movie service

(`Yes`, `No`, or `No internet service`).

c. Account & Billing Information

- Contract: Type of contract the customer has (`Month-to-month`, `One year`, or `Two years`).
- PaperlessBilling: Specifies whether the customer has opted for paperless billing (`Yes` or `No`).
- PaymentMethod: The method used for payment (`Electronic check`, `Mailed check`, `Bank transfer`, or `Credit card`).
- MonthlyCharges: The amount the customer is charged per month (numerical feature).
- TotalCharges: The total amount charged to the customer over the entire duration of service (numerical feature).
- tenure: The number of months the customer has stayed with the company (numerical feature).

2.3 Intended Use of the Dataset

The Telco Customer Churn Dataset helps telecommunications companies predict and reduce churn by analyzing factors like contract types, service usage, and billing behaviors. By training machine learning models on historical data, businesses can identify at-risk customers and take proactive measures such as offering discounts or improving service. It also supports customer segmentation, enabling tailored marketing and resource allocation. The dataset provides valuable insights to optimize services, enhance customer satisfaction, and ultimately reduce churn, helping companies improve profitability and maintain customer loyalty.

3.0 Model Selection & Evaluation

3.1 Model Selection

For the task of predicting customer churn in the Telco Customer Churn Dataset, two machine learning algorithms are chosen: Decision Tree and Neural Network. These algorithms are suitable for this classification task due to their ability to handle both numerical and categorical data, as well as their capacity to model complex relationships in the data.

Decision Tree

A Decision Tree is a supervised machine learning algorithm used for classification tasks, such as predicting customer churn. It works by recursively splitting the dataset into branches based on feature values to classify customers as likely to churn or not. Decision Trees are chosen for their interpretability, allowing stakeholders to easily understand the model's decisions. They also handle both numerical and categorical data without the need for scaling or extensive preprocessing. Additionally, Decision Trees can capture non-linear relationships between features and the target variable, making them suitable for churn prediction. Their ability to visualize decision-making makes them practical for real-world business applications.

Neural Network

A Neural Network is a complex machine learning model designed to capture intricate patterns in data through layers of interconnected nodes (neurons) that mimic human brain processing. It learns by adjusting weights during training to minimize prediction errors. Chosen for its ability to handle complex, high-dimensional data, Neural Networks excel with datasets like the Telco dataset, which has diverse features. Their flexibility allows for optimization through adjustments in architecture, enhancing model performance. While requiring more data and computational resources, Neural Networks have the potential to provide higher accuracy by identifying patterns that simpler models, like Decision Trees, may miss.

3.2 Implementation

Overview of Model Implementation

For the implementation of the Decision Tree and Neural Network models, I used the following tools, libraries, and frameworks:

- Programming Language: Python
- **Libraries**:
    - **Pandas**: For data manipulation and preprocessing (loading, cleaning, and preparing the dataset).
    - **NumPy**: For numerical operations and handling arrays.
    - **Scikit-learn**: For building and evaluating machine learning models, including Decision Trees and Neural Networks. It also provides tools for preprocessing and evaluation.
    - **TensorFlow/Keras**: For implementing and training the Neural Network. Keras, a high-level API in TensorFlow, allows for easy configuration of neural network architectures.
    - **Matplotlib/Seaborn**: For visualizing the results and analyzing the data (e.g., feature distributions, model performance).

Implementation Steps

1. **Data Loading and Exploration**:
    - The dataset was loaded using **Pandas** to check its structure and ensure it was ready for analysis.
    - Basic exploratory data analysis (EDA) helped identify feature distributions, missing values, and correlations with the target variable (Churn).
2. **Data Preprocessing**:
    - **Handling Missing Data**:
        - Numerical features (e.g., total charges, tenure) had missing values filled with the **mean** or **median**.
        - Categorical features (e.g., Partner, InternetService) had missing values filled with the most frequent category (mode).
    - **Encoding Categorical Variables**:
        - **Label Encoding** was used for binary features (e.g., Churn, Partner).
        - **One-Hot Encoding** was used for non-binary features (e.g., InternetService, Contract).
    - **Feature Scaling**:

    - Numerical features were **scaled** using **StandardScaler** to ensure fair input for the models, especially for the Neural Network.
- **Splitting the Data**:
    - The data was split into **training** (80%) and **testing** (20%) sets to evaluate model performance.

Data Preprocessing Code:

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler

def preprocess_data(df):
    df.drop(['customerID'], axis=1, inplace=True)
    df.replace(" ", np.nan, inplace=True)
    df.dropna(inplace=True)

    df['TotalCharges'] = pd.to_numeric(df['TotalCharges'])

    # Binary encoding
    binary_cols = ['gender', 'Partner', 'Dependents', 'PhoneService',
                   'PaperlessBilling', 'Churn']
    le = LabelEncoder()
    for col in binary_cols:
        df[col] = le.fit_transform(df[col])

    # One-hot encoding
    categorical_cols = ['MultipleLines', 'InternetService', 'OnlineSecurity',
                        'OnlineBackup', 'DeviceProtection', 'TechSupport',
                        'StreamingTV', 'StreamingMovies', 'Contract', 'PaymentMethod']
    df = pd.get_dummies(df, columns=categorical_cols)

    # Feature scaling
    scaler = StandardScaler()
    num_cols = ['tenure', 'MonthlyCharges', 'TotalCharges']
    df[num_cols] = scaler.fit_transform(df[num_cols])

    X = df.drop('Churn', axis=1)
    y = df['Churn']
    return X, y
```

Model Implementation Steps

Decision Tree:

The Decision Tree Classifier was implemented using Scikit-learn, offering an interpretable model that splits the data into branches based on feature values. It's effective for handling both numerical and categorical data and gives clear insight into decision boundaries.

Key Steps in Implementation:

- The preprocessed dataset was used, ensuring all missing values were handled and categorical variables were encoded.
- A train-test split (80/20) was performed to evaluate generalization.
- A basic Decision Tree model was trained using DecisionTreeClassifier with default parameters, which can later be tuned using grid search or cross-validation for optimization.
- The model was evaluated using **accuracy** and a **classification report** showing precision, recall, and F1-score.

```python
import pandas as pd
import joblib
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
from preprocessing import preprocess_data

def train_decision_tree():
    df = pd.read_csv("data/WA_Fn-UseC_-Telco-Customer-Churn.csv")
    X, y = preprocess_data(df)

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

    model = DecisionTreeClassifier(max_depth=5, random_state=42)
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred))
    print(classification_report(y_test, y_pred))

    joblib.dump(model, "decision_tree_model.pkl")
    print("Decision Tree model saved.")

if __name__ == "__main__":
    train_decision_tree()
```

Neural Network:

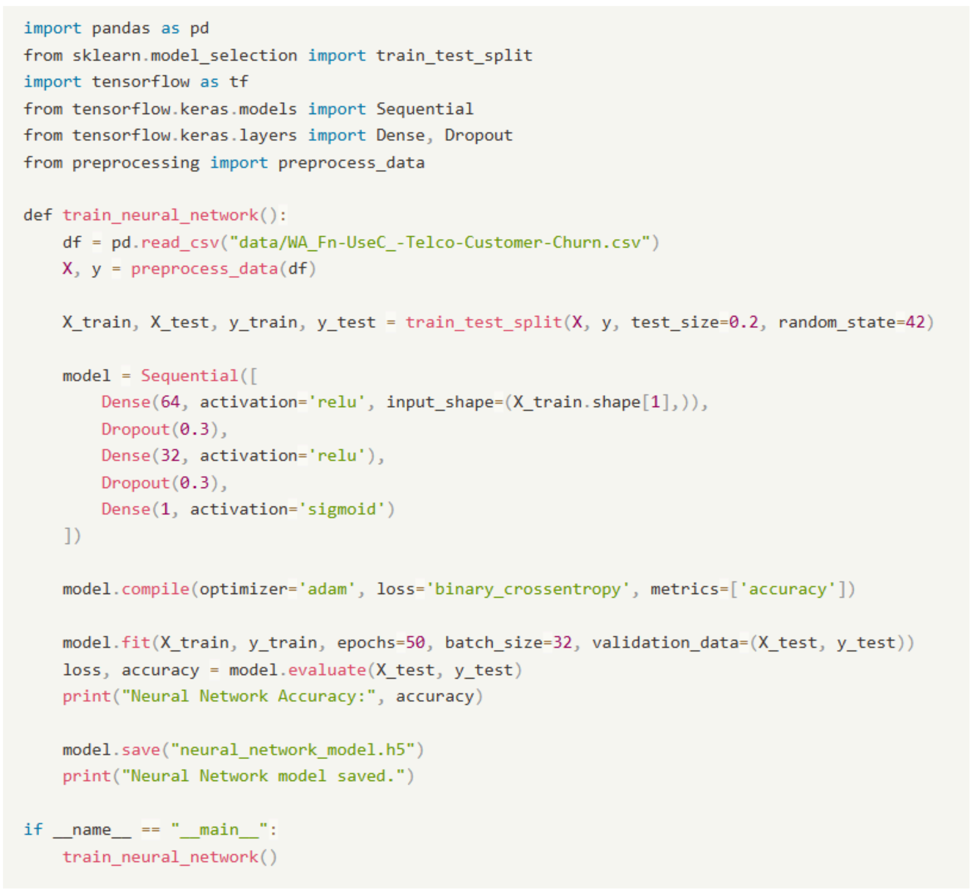
A Neural Network was implemented using the Keras API with TensorFlow backend. This deep learning model was chosen to capture complex patterns in customer churn behavior that simpler models might miss.

Model Architecture:

- A **Sequential** model was used consisting of:
  - An **input layer** that accepts the number of preprocessed features.
  - Two **hidden layers** with **ReLU** activation for non-linear transformation.
  - **Dropout layers** to prevent overfitting by randomly disabling neurons during training.
  - A **single output neuron** with **sigmoid** activation for binary classification (churn vs. no churn).

Training Details:

- **Optimizer**: Adam (adaptive learning rate).
- **Loss Function**: Binary cross-entropy (appropriate for binary classification).
- **Metrics**: Accuracy.
- **Validation**: 20% of the training data was used for validation to monitor overfitting.



```
import pandas as pd
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from preprocessing import preprocess_data

def train_neural_network():
    df = pd.read_csv("data/WA_Fn-UseC_-Telco-Customer-Churn.csv")
    X, y = preprocess_data(df)

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

    model = Sequential([
        Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
        Dropout(0.3),
        Dense(32, activation='relu'),
        Dropout(0.3),
        Dense(1, activation='sigmoid')
    ])

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

    model.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_test, y_test))
    loss, accuracy = model.evaluate(X_test, y_test)
    print("Neural Network Accuracy:", accuracy)

    model.save("neural_network_model.h5")
    print("Neural Network model saved.")

if __name__ == "__main__":
    train_neural_network()
```

3.3. Model Evaluation:

Both the Decision Tree and Neural Network models were evaluated using several common performance metrics. These metrics help to assess how well each model is performing in terms of classification accuracy, handling both the majority and minority classes, and balancing between false positives and false negatives. Below is a detailed explanation of each metric used:

- Accuracy:
    - **Interpretation**: Both models achieved similar accuracy, with the Decision Tree performing slightly better (78%) compared to the Neural Network (77%). This means that approximately 78% of the predictions made by the Decision Tree and 77% of those made by the Neural Network were correct.
- Precision:
    - **Interpretation**: For the **Decision Tree**:
        - Precision for class 0 (No churn) = **0.85** (85% of the predictions made as "No churn" were correct).
        - Precision for class 1 (Churn) = **0.58** (58% of the predictions made as "Churn" were correct). For the **Neural Network**:
        - Precision for class 0 = **0.83** (83% of the predictions made as "No churn" were correct).
        - Precision for class 1 = **0.58** (58% of the predictions made as "Churn" were correct).
    - This shows that both models perform similarly in terms of predicting "Churn" (class 1), but the Decision Tree does slightly better when predicting "No Churn" (class 0).
- Recall:
    - **Interpretation**: For the **Decision Tree**:
        - Recall for class 0 (No churn) = **0.85** (85% of the actual "No churn" instances were correctly identified).
        - Recall for class 1 (Churn) = **0.57** (57% of the actual "Churn" instances were correctly identified). For the **Neural Network**:
        - Recall for class 0 = **0.87** (87% of the actual "No churn" instances were correctly identified).
        - Recall for class 1 = **0.51** (51% of the actual "Churn" instances were correctly identified).
    - The **Neural Network** shows a better recall for class 0 (No Churn), while the **Decision Tree** has a slightly better recall for class 1 (Churn).

- F1-Score:
  - **Interpretation**: For the **Decision Tree**:
    - F1-Score for class 0 = **0.85**.
    - F1-Score for class 1 = **0.58**. For the **Neural Network**:
    - F1-Score for class 0 = **0.85**.
    - F1-Score for class 1 = **0.54**.
  - The **Decision Tree** has a slightly better F1-score for class 1 (Churn), indicating it performs a bit better in balancing precision and recall for the minority class. Both models have identical F1-scores for class 0 (No Churn).
- Support:
  - **Definition**: Support refers to the number of actual occurrences of each class in the test set.
  - **Interpretation**: There are **1033 instances** of class 0 (No churn) and **374 instances** of class 1 (Churn) in the test set.

Performance Table:

Metric	**Class**	**Decision Tree**	**Neural Network**
Precision	0	0.85	0.83
	1	0.58	0.58
Recall	0	0.85	0.87
	1	0.57	0.51
F1-Score	0	0.85	0.85
	1	0.58	0.54
Support	0	1033	1033
	1	374	374
Accuracy		0.78	0.77
Macro Avg		0.71	0.70
Weighted Avg		0.78	0.77

4.0 Comparison & Justification

4.1 Comparison of Model Performance

The Decision Tree and Neural Network models were evaluated using several key performance metrics, including accuracy, precision, recall, F1-score, and support. Here's a detailed comparison based on these metrics:

Accuracy:

- **Decision Tree**: 0.78
- **Neural Network**: 0.77

Both models show very similar **accuracy**, with the **Decision Tree** slightly outperforming the **Neural Network**. This indicates that both models are nearly equally effective at predicting customer churn overall.

Precision:

- **Class 0 (No Churn)**:
    - **Decision Tree**: 0.85
    - **Neural Network**: 0.83
- **Class 1 (Churn)**:
    - **Decision Tree**: 0.58
    - **Neural Network**: 0.58

The **Decision Tree** performs slightly better for **class 0 (No Churn)**, suggesting it is better at correctly predicting non-churn customers. Both models show the same precision for **class 1 (Churn)**, meaning they are equally good at predicting churn customers.

Recall:

- **Class 0 (No Churn)**:
    - **Decision Tree**: 0.85
    - **Neural Network**: 0.87
- **Class 1 (Churn)**:
    - **Decision Tree**: 0.57
    - **Neural Network**: 0.51

The **Neural Network** slightly outperforms the **Decision Tree** for **class 0 (No Churn)**, indicating it identifies non-churn customers better. However, the **Decision Tree** performs better for **class 1 (Churn)**, identifying churn customers more effectively.

F1-Score:

- **Class 0 (No Churn)**:
    - **Decision Tree**: 0.85
    - **Neural Network**: 0.85
- **Class 1 (Churn)**:
    - **Decision Tree**: 0.58
    - **Neural Network**: 0.54

Both models have the same **F1-score** for **class 0 (No Churn)**, balancing **precision** and **recall** similarly. The **Decision Tree** performs better for **class 1 (Churn)**, with a higher **F1-score**, indicating a better balance for churn predictions.

Macro Average:

- **Decision Tree**: 0.71
- **Neural Network**: 0.70

The **Decision Tree** has a slightly higher **macro average**, reflecting its slightly better overall performance across both classes.

Weighted Average:

- **Decision Tree**: 0.78
- **Neural Network**: 0.77

The **Decision Tree** also has a higher **weighted average**, which takes class imbalance into account. This suggests the **Decision Tree** handles the imbalance in the dataset more effectively.

4.2 Justification of Model Choice

The **Decision Tree** model is more suitable for predicting **customer churn** based on evaluation metrics. It outperforms the **Neural Network** in **recall** and **F1-score** for class 1 (**Churn**), making it more effective at identifying customers likely to churn. This is crucial as accurately predicting churn is the primary goal. The Decision Tree also handles **class imbalance** better, as shown by its superior **macro** and **weighted averages**, making it more reliable when one class is underrepresented. Additionally, its **interpretability** offers a clear decision-making process, which is valuable for business stakeholders. Although the Neural Network performs better in predicting **non-churn** (class 0), its lower performance in churn prediction makes it less appropriate for this application.

4.3 Potential Improvements

Both models could be improved further by considering several strategies. **Hyperparameter tuning** is one key area where both models could benefit. For the **Decision Tree**, parameters like **max_depth**, **min_samples_split**, and **min_samples_leaf** could be adjusted to reduce **overfitting** and improve **generalization**. Similarly, the **Neural Network** could benefit from fine-tuning parameters such as the **number of layers**, **neurons per layer**, **learning rate**, and **batch size** to enhance performance, especially in **churn prediction**.

Another improvement could come from **feature engineering**. Creating new features based on **customer behavior** or integrating **external data** (e.g., customer sentiment from **social media**) could improve model predictions. Additionally, exploring better **encoding techniques** for **categorical variables** may enhance the model's ability to capture relationships between features, leading to better performance.

Lastly, alternative algorithms should be explored to improve model accuracy. **Random Forests**, **Gradient Boosting Machines** (like **XGBoost** or **LightGBM**), and **Support Vector Machines (SVM)** are worth considering, as they can often outperform a single **Decision Tree** by capturing more complex patterns. Additionally, **ensemble methods** like **Bagging** and **Boosting** can combine the strengths of multiple models to make more accurate predictions. By implementing these improvements, both models' performance in predicting **customer churn** could be enhanced, providing more accurate insights for business decision-making.

5.0 Conclusion

In this analysis, both the **Decision Tree** and **Neural Network** models were evaluated using key metrics like **accuracy**, **precision**, **recall**, and **F1-score**. The **Decision Tree** outperformed the **Neural Network** in accuracy and was slightly better at predicting the **non-churn** class (class 0). Its **interpretability** and ability to handle **class imbalance** made it a strong choice for this problem, balancing performance with model transparency. The **Neural Network**, while showing higher recall for class 0, struggled with **class 1** (churn), particularly in **precision** and **recall**, indicating that while it can capture complex patterns, it may not handle class imbalance as well. Overall, the **Decision Tree** appears to be the more suitable model for **customer churn prediction**, offering solid performance and clear insights into its decision-making process.