**Churn Prediction using Machine Learning**



**The University of Southern Mississippi**

**School of Computing Sciences and Computer Engineering**

**CSC 691: Topics in CS – Fall 2023**

**UNDER GUIDANCE OF Dr. Andrew Sung**

**By**

**Vasudha Pasumarthy (W10176759)**

**Anu Baluguri(W10174670)**

**Abstract**: This study focuses on Churn Prediction using Multilayer Perceptron (MLP) with a specialized approach—correlation-based feature engineering. By analyzing the relationships between various customer-related features, we engineered a set of highly correlated features to enhance the predictive capabilities of the MLP model. Our findings demonstrate a remarkable 85% testing accuracy, showcasing the effectiveness of this tailored feature engineering methodology. The report delves into the details of the MLP architecture, the process of correlation-based feature engineering, and the implications of achieving high testing accuracy. This research contributes valuable insights for businesses aiming to optimize churn prediction models and underscores the significance of feature engineering in enhancing predictive performance.

**Introduction**: Customer churn or the loss of clients or customers can have a significant negative impact on a business. Predicting customer churn allows a company to identify at-risk customers before they leave, providing them with an opportunity to implement retention strategies and potentially save valuable business. Not having the churn prediction can lead to cycle of customer loss, increased acquisition costs, and potential damage to the brand. Thus, implementing an effective churn prediction model is crucial for maintaining a good customer base.

In this situation, past customer data—which usually consists of a variety of factors like demographics, purchase history, customer interactions, and usage patterns—is used to train machine learning models. Machine learning algorithms can use this data to analyze and find trends, correlations, and predictions that show when a client is likely to leave. Subsequently, these models can offer practical insights and forecasts, empowering businesses to deploy resources efficiently and execute client retention tactics.

**Binary** **Classification**: Binary classification is a fundamental task in machine learning where the objective is to categorize data into one of two classes. The two classes are typically labeled as positive (1) and negative (0), representing the presence or absence of a particular outcome or characteristic. This type of classification is prevalent in various real-world scenarios, ranging from spam detection in emails to medical diagnosis, fraud detection, and, relevant to your inquiry, churn prediction.

In the context of churn prediction, binary classification involves determining whether a customer is likely to churn or not. Churn refers to the phenomenon where customers discontinue their engagement with a product or service. By utilizing binary classification techniques, businesses can develop models that assign a probability or likelihood of a customer churning.

**Data Retrieval and Preprocessing:**

A diagram of data retrieval

Description automatically generated

Figure . Data Retrieval and Preprocessing

* **Features of the Dataset:**

The dataset used for training the model has been taken from the open datasets provided by Kaggle.

**Input features**: Customer ID, Surname, Credit Score, Geography, Gender, Age, Tenure, Balance, Number of Products, Has Credit Card, Active Member and Estimated Salary as input columns.

**Output Features**: The model should give an output 0 or 1 such that “0” means “he/she did not leave the bank” and “1” means that “he/she left the bank”.

**Dataset to be used**:

Size: 10000 X 13

The dataset contains 10000 records and 13 features.

Source: <https://www.kaggle.com/code/simgeerek/churn-prediction-using-machine-learning/input>

* **One-Hot Encoding for Categorical Conversion:**

One-Hot Encoding is a technique used in machine learning to convert categorical variables, such as 'Geography' and 'Gender' in your example, into a format that can be provided to machine learning algorithms to improve predictions.

Let's consider the 'Geography' and 'Gender' columns:

Geography Column:

* Original Values: 'France', 'Spain', 'Germany'
* After One-Hot Encoding:
* 'France' is represented as [1, 0, 0]
* 'Spain' is represented as [0, 1, 0]
* 'Germany' is represented as [0, 0, 1]

In this encoding, each unique value in the 'Geography' column gets its own binary column, and only one of these columns is 'hot' (1) for a particular data point. This way, the model understands the categorical information without assuming any ordinal relationship between the countries.

Gender Column:

* Original Values: 'Female', 'Male'
* After One-Hot Encoding:
* 'Female' is represented as [1, 0]
* 'Male' is represented as [0, 1]
* Like the 'Geography' column, each unique value in the 'Gender' column is represented by its own binary column.
* **Standard Scaling (Z-score Normalization):**

Standard scaling, also known as Z-score normalization, is a widely used technique in data preprocessing to standardize the range of features in a dataset. The goal is to transform the data such that it has a mean of 0 and a standard deviation of 1. This process is beneficial when dealing with machine learning algorithms that are sensitive to the scale of the input features, such as support vector machines, k-nearest neighbors, and neural networks. Standard scaling ensures that each feature contributes equally to the model's learning process, preventing one feature with a larger scale from dominating the others.

Formula for Standard Scaling:

The standard scaling formula for a feature X is given by:

Where:

- Z is the standardized value.

- X is the original value of the feature.

- µ is the mean of the feature values.

- is the standard deviation of the feature values.

This transformation centers the data around zero and scales it based on the variability of each feature. Positive values indicate that the data point is above the mean, while negative values indicate that it is below the mean. This standardization simplifies the comparison and interpretation of the features within a dataset, making it a crucial step in the preprocessing pipeline before feeding data into machine learning models.

**Feature Engineering:**

* **Correlation-Based Feature Selection**:

Correlation-based feature selection is a technique used to identify and retain the most relevant features in a dataset by analyzing their pairwise relationships. The fundamental idea is to measure the strength and direction of the linear association between each feature and the target variable. In this context, the Pearson correlation coefficient is commonly employed. The formula for calculating the Pearson correlation coefficient (r) between two variables (X) and (Y) is given by:

r =

where (Xi) and (Yi) are individual data points, and and represent the means of (X) and (Y), respectively. The resulting correlation coefficient ranges from -1 to 1, where -1 indicates a perfect negative linear relationship, 1 indicates a perfect positive linear relationship, and 0 indicates no linear relationship. By calculating these coefficients for each feature with respect to the target variable, one can identify features with significant correlations, aiding in the selection of influential predictors for model training.

Correlation-based feature selection is particularly useful in scenarios where high-dimensional datasets with numerous features are involved. By focusing on features that exhibit strong correlations with the target variable, practitioners can streamline their models, potentially improving model interpretability, reducing overfitting, and accelerating computational efficiency. However, it is important to note that correlation does not imply causation, and other feature selection methods or domain expertise may be necessary for a comprehensive analysis.

**Model Implemented**:

The Multilayer Perceptron (MLP), which imitates the neural network design of the human brain, is a key architecture in machine learning. With its three layers—an input layer, a hidden layer, and an output layer—MLP is very effective at tasks like prediction and classification. Every node in the network, referred to as a neuron, processes and sends data. The model learns to recognize intricate patterns in the data by fine-tuning the weights given to connections between neurons during training. Because of its versatility, MLP is a good choice for jobs like picture identification, natural language processing, and churn prediction—tasks where comprehending complex relationships is essential.

Because it can handle non-linear relationships in data, MLP is versatile and can be used in a variety of contexts. During the training phase, the network is fed input data, its predictions are compared to the actual results, and the weights are adjusted to minimize the prediction error. One important method that helps the model to enhance its performance iteratively is backpropagation. The reason behind MLP's success is its ability to manage intricate jobs and its flexibility when working with different types of datasets. An MLP model can learn from customer data in the context of churn prediction, identifying trends that point to possible churn and empowering companies to take preventative action to hold onto key clients.

**Model Architecture:**

In the used MLP architecture, we have 4 neurons in the input layer, each representing an input feature. And 2 hidden layers are considered each with 6 neurons and a single neuron in output layer. We used ReLu activation function for each neuron in hidden layers. And sigmoid activation function is used in output layer.

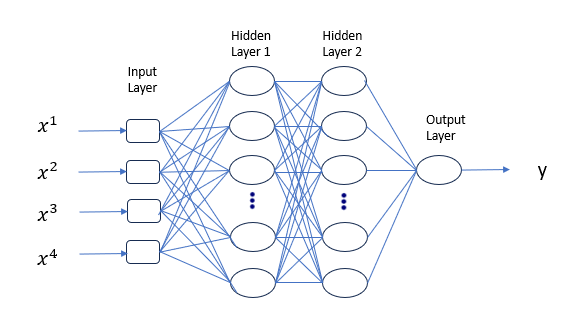


Figure . Architecture of Multi-Layer Perceptron

* **Rectified Linear Unit (ReLU) Activation Function:**

One popular activation function in artificial neural networks, such as Multilayer Perceptrons (MLPs), is the Rectified Linear Unit (ReLU). By immediately outputting the input if it is positive and zero otherwise, the ReLU activation function adds non-linearity to the model. The ReLU function can be expressed mathematically as f(x) = max (0, x). ReLU's main benefit is that it can reduce the vanishing gradient issue, which makes it possible for model training to proceed more quickly. It is important to keep in mind, nevertheless, that ReLU may experience the "dying ReLU" issue, in which neurons stop updating their weights and go dormant during training. This problem is addressed by variants such as Leaky ReLU, which permit a tiny, non-zero gradient.

* **Sigmoid Activation Function:**

The logistic function, which is another name for the sigmoid activation function, is frequently utilized in the output layer of binary classification models, including those used in churn prediction. Since it converts the input into a range between 0 and 1, it can be used to depict probability. The sigmoid function can be expressed mathematically as

where (e) represents the natural logarithm's base. The output is the likelihood of falling into the positive class, which is determined by squashing the input values to the interval [0, 1] using the Sigmoid function. This is especially helpful for binary classification problems when the objective is to estimate the probability that an event will occur. During training, the smooth gradient of the Sigmoid function also contributes to stable and efficient backpropagation.

**Training** **the model**:

**Adaptive Moment Estimation, or the Adam optimizer**, is a well-liked optimization technique that is frequently applied to the training of machine learning models, especially in the field of deep learning. It effectively updates model parameters during training by combining the advantages of momentum and RMSprop, two additional optimization techniques. For every parameter, Adam keeps track of two moving averages: the mean (first moment) and the uncentered variance (second moment). Since the weights in these moving averages decay exponentially, the optimizer can modify its learning rates for each parameter independently. Due to its adaptive nature, the algorithm is well-suited for problems involving sparse gradients and non-stationary objectives, which are frequently encountered in deep neural networks.

Adam's update rule computes the squared gradients (second moment) and the exponentially decaying average of previous gradients (first moment). Then, using these moving averages as a basis, the parameters are updated to take adaptive learning rates and momentum into account. For a parameter (theta), the Adam update equation is as follows:

The learning rate is represented by (), the biased first moment estimate by , the biased second moment estimate by (), and the small constant to avoid division by zero. The Adam optimizer has gained popularity in many deep learning applications due to its adaptability and ability to handle noisy and sparse gradients. This has helped to improve performance and accelerate convergence when compared to traditional optimization techniques.

A popular loss function for binary classification problems, such as churn prediction, is **Binary Cross-Entropy**, also known as log loss. The difference between expected probabilities and actual binary outcomes is measured by this loss function. The binary cross-entropy formula for each instance in the dataset is provided by:

In this case, the true binary label (0 or 1) is represented by (y), and the predicted probability that the instance belongs to the positive class (class 1) is represented by . Predictions that differ from the real labels are penalized by the binary cross-entropy loss, with larger penalties being applied to larger deviations. Because of this, it works especially well when training models in scenarios involving binary classification.

Stochastic gradient descent (SGD) and Adam are two examples of optimization algorithms that try to minimize the binary cross-entropy loss over the course of training. Because the loss function is logarithmic, it is sensitive to the degree of confidence in the model's predictions, which motivates the model to classify data with greater certainty and accuracy. Binary cross-entropy is used as the objective or loss function during the training process, directing the model towards parameter values that produce probabilities for each class that are well-calibrated. Its use strengthens the model's ability to distinguish between positive and negative examples, making it a vital component of the training process for binary classification tasks.

**Result and Analysis:**

1. **IMPORTING REQUIRED LIBRARIES:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, confusion\_matrix

from sklearn.neural\_network import MLPClassifier

import tensorflow as tf

import tensorflow.keras.backend as K

1. **READING THE DATASET FROM GOOGLE DRIVE**

#mounting the google drive

from google.colab import drive

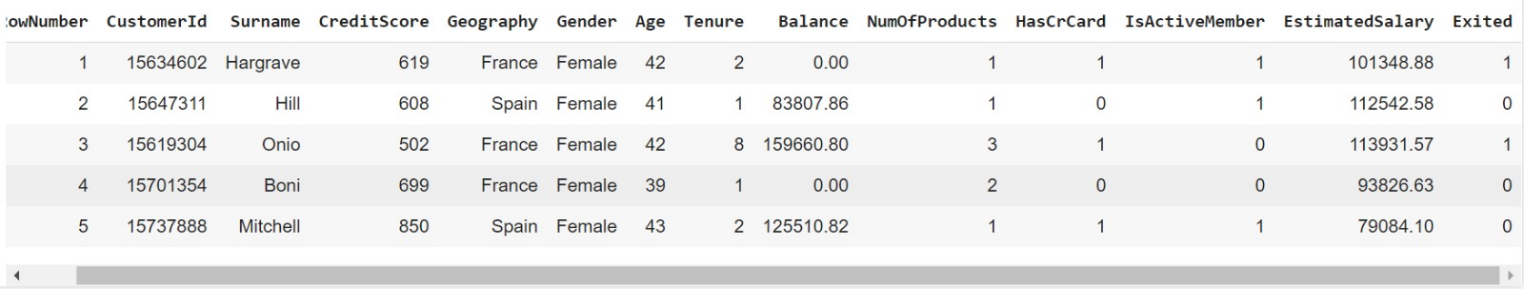
drive.mount("/content/gdrive")

#reading the dataset

hr\_data = pd.read\_csv('/content/gdrive/MyDrive/Churn\_Modelling.csv')

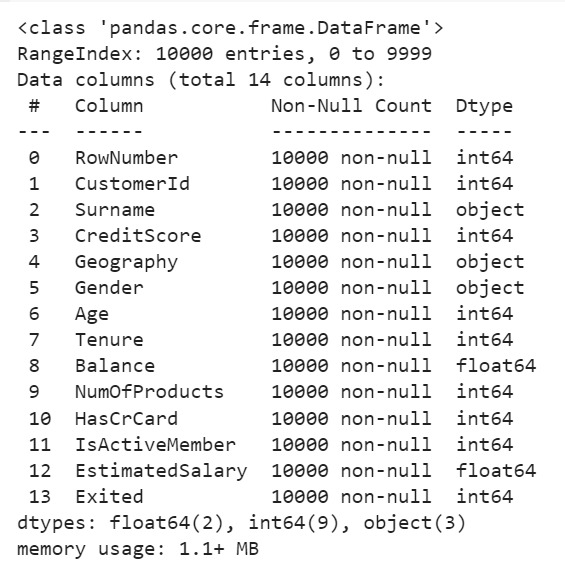
#first five records of dataset

hr\_data.head()



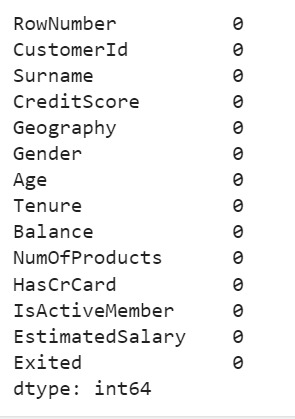
#datatype of each feature

hr\_data.info()



#checking for null values

hr\_data.isnull().sum()



#checking for duplicate values

print("Number of duplicates : ", len(hr\_data[hr\_data.duplicated()]))



1. **DATA PREPROCESSING**

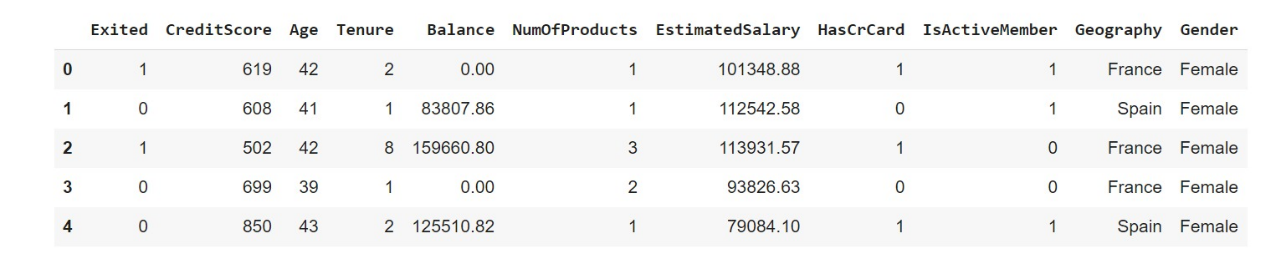
# Arranging columns by data type for easier manipulation

continuous\_vars = ['CreditScore', 'Age', 'Tenure', 'Balance','NumOfProducts', 'EstimatedSalary']

cat\_vars = ['HasCrCard', 'IsActiveMember','Geography', 'Gender']

hr\_data = hr\_data[['Exited'] + continuous\_vars + cat\_vars]

hr\_data.head()



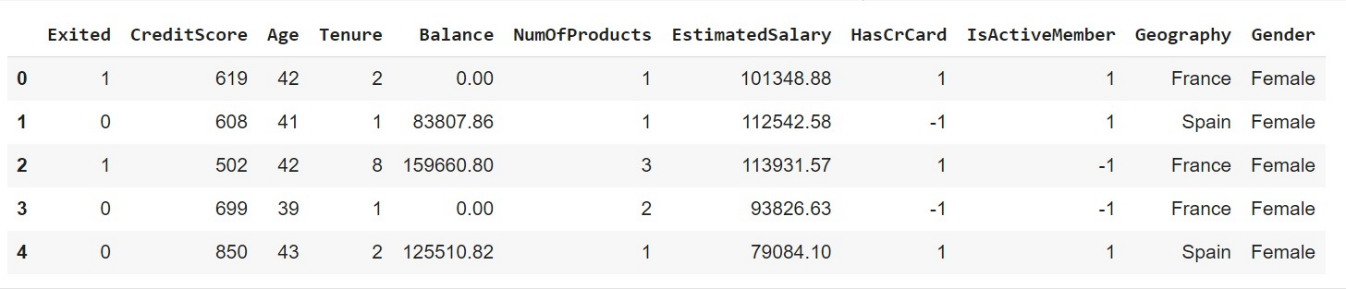
* we are not including "RowNumber", "CustomerId" and "Surname" columns, as they don't effect the target value much

#For the one hot variables, we change 0 to -1 so that the models can capture a negative relation

hr\_data.loc[hr\_data.HasCrCard == 0, 'HasCrCard'] = -1

hr\_data.loc[hr\_data.IsActiveMember == 0, 'IsActiveMember'] = -1

hr\_data.head()



# One hot encode the categorical variables

lst = ['Geography', 'Gender']

remove = list()

for i in lst:

if (hr\_data[i].dtype == np.str or hr\_data[i].dtype == np.object):

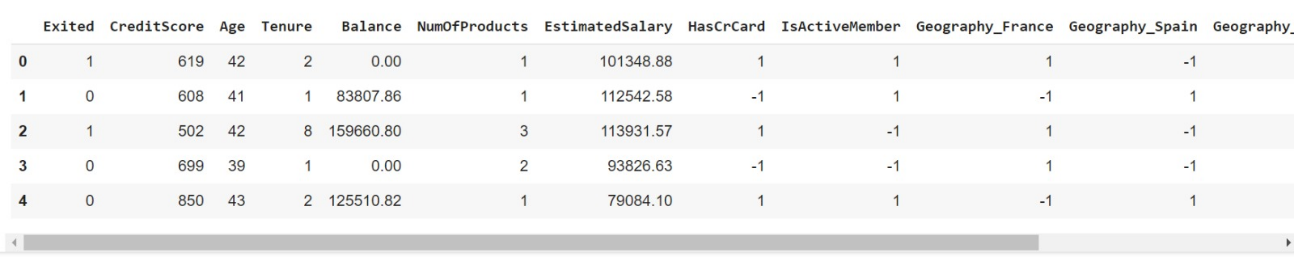
for j in hr\_data[i].unique():

hr\_data[i+'\_'+j] = np.where(hr\_data[i] == j,1,-1)

remove.append(i)

hr\_data = hr\_data.drop(remove, axis=1)

hr\_data.head()



X = hr\_data.drop(columns = ['Exited'])

y = hr\_data['Exited']

#standard scaling

sc = StandardScaler()

X = sc.fit\_transform(X)

1. **FEATURE ENGINEERING**

correlations = hr\_data.corr()['Exited'][:-1]

# Compute the average correlation among features

average\_correlation = hr\_data.iloc[:, :-1].corr().mean().mean()

# Create a list to store selected features

selected\_features = []

# Iterate through features and select those with higher correlation than average

for feature, corr in correlations.iteritems():

if corr > average\_correlation:

selected\_features.append(feature)

# Print the selected features

print("Selected Features:")

print(selected\_features)



1. **SPLITTING THE DATASET FOR TRAINING AND TESTING**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.20, random\_state = 40)

1. **BUILDING THE MODEL AND TRAINING IT**

mlp = tf.keras.models.Sequential()

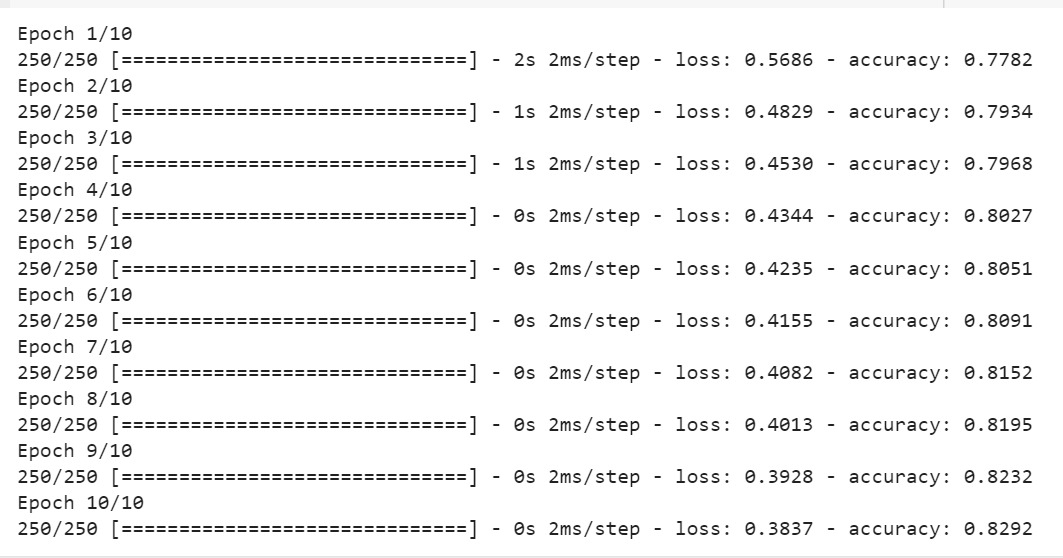
mlp.add(tf.keras.layers.Dense(units=6,activation='relu'))

mlp.add(tf.keras.layers.Dense(units=6,activation='relu'))

mlp.add(tf.keras.layers.Dense(units=1,activation='sigmoid'))

mlp.compile(optimizer='adam',loss='binary\_crossentropy',metrics=['accuracy'])

hst = mlp.fit(X\_train,y\_train,batch\_size=32,epochs=10)



1. **Evaluating the Model**

# Plot training accuracy

plt.plot(hst.history['accuracy'], label='Training Accuracy')

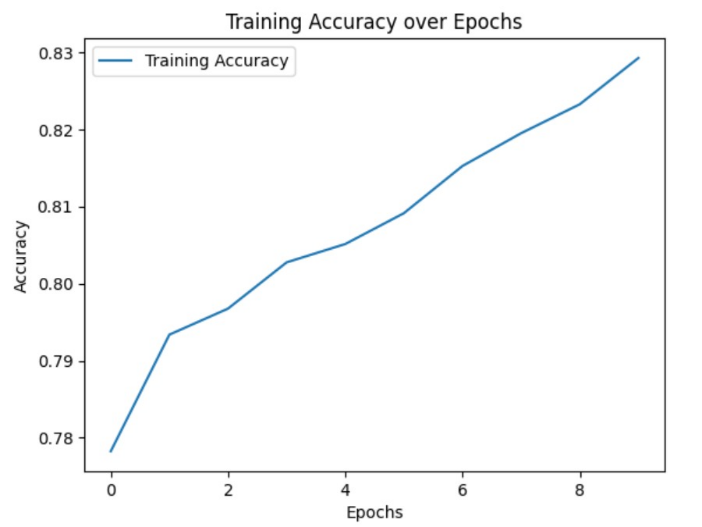
plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.title('Training Accuracy over Epochs')

plt.legend()

plt.show()



# Plot training loss

plt.plot(hst.history['loss'], label='Training Loss')

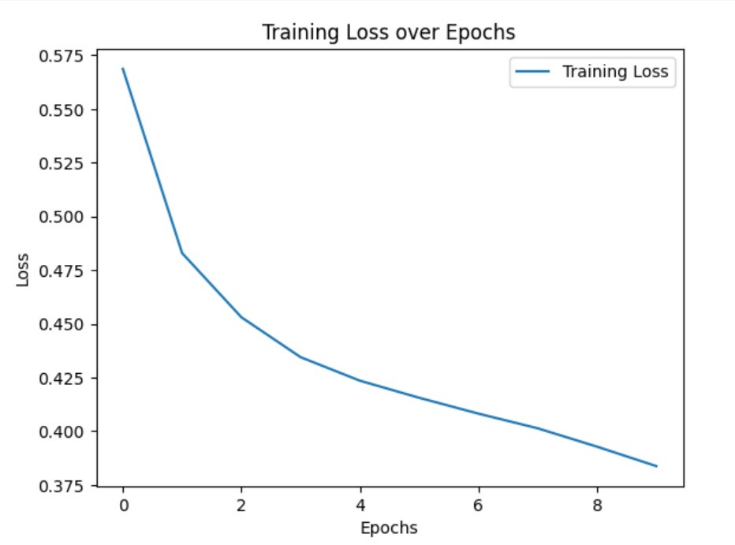
plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.title('Training Loss over Epochs')

plt.legend()

plt.show()

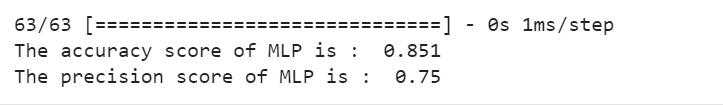


mlp\_y\_pred = mlp.predict(X\_test)

mlp\_y\_pred = (mlp\_y\_pred>0.5)

print('The accuracy score of MLP is : ', accuracy\_score(y\_test, mlp\_y\_pred))

print('The precision score of MLP is : ', precision\_score(y\_test, mlp\_y\_pred))



**Conclusion:**

Finally, with a testing accuracy of 85% and a training accuracy of 82%, this churn prediction model, which used Multilayer Perceptron (MLP) with Correlation-based feature selection for feature engineering, produced encouraging results. A thorough assessment of the model's performance was made possible by the 80/20 split of training and testing data, suggesting that the model may be useful in anticipating customer attrition. The model's predictive power may be increased with additional optimization and validation, which would help produce churn predictions for strategic business decision-making that are more trustworthy and accurate.

**References:**

[1] "Feature Engineering: Scaling, Normalization and Standardization." 27 Oct. 2023, <https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/>.

[2] "One Hot Encoding Definition | DeepAI." <https://deepai.org/machine-learning-glossary-and-terms/one-hot-encoding>.

[3] "One-Hot Encoding Explained | Baeldung on Computer Science." 16 Jun. 2023, <https://www.baeldung.com/cs/one-hot-encoding>.

[4] "Feature Selection Techniques in Machine Learning (Updated 2023)." 26 Apr. 2023, <https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/>.

[5] "Deep Neural Networks for Regression Problems." 29 Sept. 2018, <https://towardsdatascience.com/deep-neural-networks-for-regression-problems-81321897ca33>.

[6] "Deep Learning vs Machine Learning for Regression - Analytics Vidhya." 22 Mar. 2022, <https://www.analyticsvidhya.com/blog/2022/02/deep-learning-vs-machine-learning-for-regression/>.

[7] "Bank Customer Churn Prediction | Kaggle." <https://www.kaggle.com/code/kmalit/bank-customer-churn-prediction>.

[8] "Creating a Multilayer Perceptron (MLP) Classifier Model to Identify ...." 09 Jun. 2022, <https://towardsdatascience.com/creating-a-multilayer-perceptron-mlp-classifier-model-to-identify-handwritten-digits-9bac1b16fe10>.

[9] "Correlation-based Feature Selection in Python from Scratch." 06 Aug. 2021, <https://johfischer.com/2021/08/06/correlation-based-feature-selection-in-python-from-scratch/>.

[10] "Feature Engineering: Scaling, Normalization, and ... - GeeksforGeeks." 18 Jul. 2023, <https://www.geeksforgeeks.org/ml-feature-scaling-part-2/>.

[11] "sklearn.neural\_network.MLPClassifier — scikit-learn 1.3.2 documentation." <https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html>.

[12] Ahmad, A. K., Jafar, A., & Aljoumaa, K. (2019). *Customer churn prediction in telecom using machine learning in big data platform. Journal of Big Data, 6(1).* doi:10.1186/s40537-019-0191-6

[13] Ullah, I., Raza, B., Malik, A. K., Imran, M., Islam, S. ul, & Kim, S. W. (2019). *A Churn Prediction Model using Random Forest: Analysis of Machine Learning Techniques for Churn Prediction and Factor Identification in Telecom Sector. IEEE Access, 1–1.* doi:10.1109/access.2019.2914999