SAINT LOUIS UNIVERSITY™

Introduction to Machine Learning (CSCI-4750-02)

Dr. Jie Hou

Author's:

1.Jyostna Akhil paila

2.Mani Teja Akinapalli

3.Shiva Narayana Githa

Title: Warfarin dose Prediction using Machine Learning

INTRODUCTION

Machine learning (ML) is one of a significant and promising technology used in different fields and applications. This report provides a detailed information on warfarin dose prediction for a patient based on different conditions. Warfarin is an effective anticoagulant used to treat thromboembolism. For patients suffering with thromboembolism, mostly the warfarin dose is prescribed inaccurately by the doctors which may cause critical clinical issues. Machine Learning can assist doctors in determining the correct warfarin dosage to prevent thromboembolism and lower the risk of bleeding in patients. ML has been widely employed in medical studies, including the dose of warfarin. ML methods such as random forest, support vector machines, and artificial neural networks have been used to evaluate warfarin dosage data and accurately forecast warfarin dose. This paper focuses on the use of ML for warfarin dosage. We will talk about current warfarin dosage research, the data utilized for ML prediction, the various ML algorithms used for warfarin dosing, and the problems and limits of ML for warfarin dosing. We will also talk about the possible advantages of ML-based warfarin dosage prediction.

This report will be split into four key components. The first section will offer an overview of warfarin dosage, emphasizing its relevance and the difficulties connected with it. The second section will go into the data utilized for ML-based warfarin dosage, the various ML algorithms used for prediction, and the outcomes of the various experiments. The final section will go through the possible advantages of ML-based warfarin dosage prediction. The fourth and last

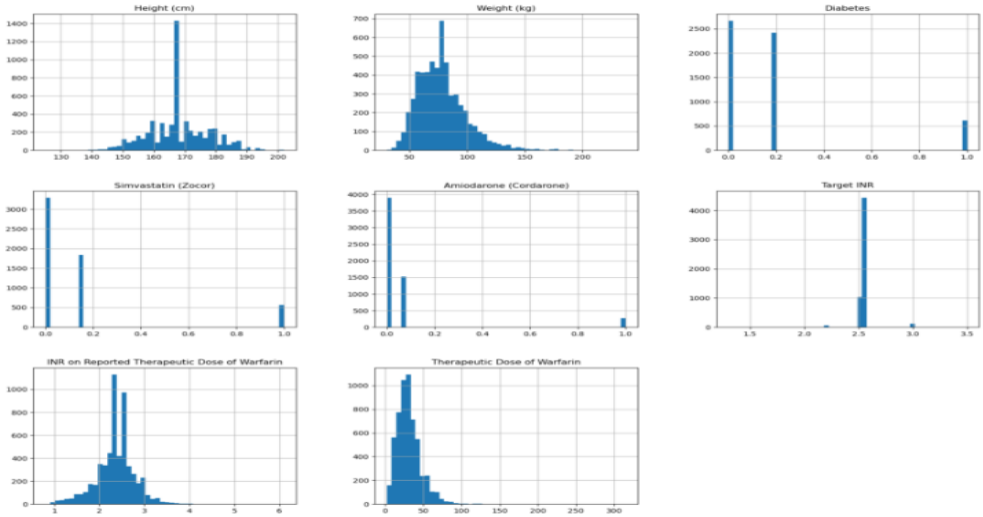
part will explore the problems and limits of ML for warfarin dose and the potential topics for further study.

The consortium's major purpose is to uncover genetic variations that impact warfarin dosage needs and to build a warfarin dosing algorithm based on such variants. The methodology is designed to assist healthcare practitioners in prescribing the best warfarin dosage for each patient. The collaboration has also performed research on other pharmacogenetic problems linked to warfarin, such as drug-drug interactions and the effect of race and ethnicity on warfarin dosage needs.

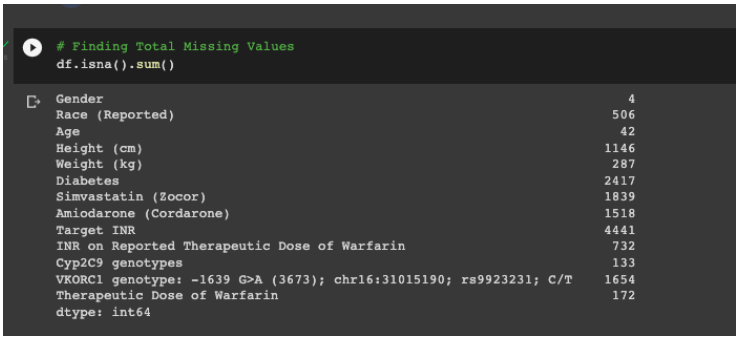
The International Warfarin Pharmacogenetics Consortium (IWPC) developed an algorithm to predict the warfarin dose based on clinical and pharmacogenetic data. To compute a basic warfarin dose, the algorithm considers a patient's age, gender, body weight, height, race, and medical history. The algorithm then considers genetic variation in CYP2C9 and VKORC1, two genes involved in warfarin metabolism, to calculate a genetically adjusted warfarin dosage. Other variables, such as concurrent drugs, that might alter warfarin metabolism, are also taken into account by the algorithm. The algorithm was created and tested using a dataset of over 8,000 patients of diverse ethnicities, and it has been demonstrated to be more accurate than previous approaches in estimating warfarin dose. As a result, it is a valuable tool for physicians in identifying the appropriate warfarin dose for their patients.

Dataset

The has been taken from pharmGKB, it is one of the best online resource in the field of medicine and provides data based on how human genetic variation affects response to medications. Initially there are many feature available in the dataset but we have taken some of the important features, which are - Gender, Race, Age, Height (cm), Weight (kg), Diabetes, Simvastatin (Zocor), Amiodarone (Cordarone), Target INR, INR on Reported Therapeutic Dose of Warfarin, Cyp2C9 genotypes, VKORC1 genotype: -1639 G>A (3673); chr16:31015190; rs9923231; C/T. Target feature as Therapeutic Dose of Warfarin.



Initially there are lot of missing values in each feature. This is little challenging as if we remove/drop the missing value row then there will be lot of rows removed. So we used mean and mode to replace the missing value in the feature.



```
# Finding Total Missing Values
df.isna().sum()
```

```
Gender                                                                4
Race (Reported)                                                     506
Age                                                                  42
Height (cm)                                                        1146
Weight (kg)                                                         287
Diabetes                                                           2417
Simvastatin (Zocor)                                                1839
Amiodarone (Cordarone)                                             1518
Target INR                                                         4441
INR on Reported Therapeutic Dose of Warfarin                        732
Cyp2C9 genotypes                                                    133
VKORC1 genotype: -1639 G>A (3673); chr16:31015190; rs9923231; C/T  1654
Therapeutic Dose of Warfarin                                        172
dtype: int64
```

Missing Values

For Categorical features we used mode to replace missing values and for numerical feature we used mean to replace the missing values.

Methods

The project total overview flowchart indicates the pipeline of the project from the dataset download to the evaluation metrics. First, we used an open source web link https://www.pharmgkb.org/downloads and downloaded "IWPC Data Set" and loaded in Goggle Colab.

1. Collect and explore data:

   - Collect data from PharmGKB, IWPC Data Set
   - Explore data to gain insights

2. Analyse the dataset.

   - Check of missing values and others.
   - Replace the missing values with mode and mean of the feature.

     Mode for Categorical values and Mean for Continues/Numerical values.

3. Prepare the data for the machine learning model.

   - Split data into train and test sets.
   - Transformation for categorical features
   - Normalization of train and test sets
   - Converting the target feature into binary

4. Choose a model:

   - Select an appropriate ML model based on data.

   - Train the model

   - Visualize the output.

5. Evaluate the model:

   - Evaluate the model on the test set

   - Compare model performance against baselines and other models

6. Deploy the model:

   - Deploy the model to production

   - Observe the model performance in testing data

7. Improve the model:

   - Analyse model performance data

   - Iterate on the model and repeat the process

The data set was collected from an open source. First, the data set is read into the project file and reviewed. The missing values of each feature are then detected and filled with the mode and mean values from the existing data set for that feature. Following that, the data set is structured and cleaned to ensure that it is in useable shape. Finally, the data is displayed to gain a better understanding of the data set and to detect any connections and trends. The data was then represented using bar graphs and pie charts.

After Pre-processing the data, We need to split the data into training and testing sets to feed the model and need to do normalization and transformation for the data. For categorical values we need to do the transformation (i.e., For Gender - Male – 1 and Female - 2) and once it's done. we need to Min-Max normalization to convert the large numerical data to 0-1.

Also We need to convert the target column into binary data. (For Classification if the Therapeutic Dose of Warfarin > 0 then 1 and if it''s <= 0 the 1)

The project's purpose is to create a machine-learning algorithm that can estimate warfarin dose reliably. We will employ a dataset of patients with various characteristics, such as age, gender, race, and medical history, to do this. The data will then be used to develop a model that predicts the best warfarin dose for each patient. The model will then be evaluated on a different dataset. To build the model, we will employ a number of machine learning methods, including KNN, decision trees, random forests, and support vector machines. We will also analyse the model using measures such as accuracy, precision, and recall. Finally, we will talk about the consequences of our findings and the model's possible applications.

The correlation between the independent variables in the data set is used to determine multicollinearity. If two or more of the independent variables are substantially correlated, we can conclude that the data is multicollinear. We may analyse multicollinearity using statistical approaches such as a correlation matrix, a variance inflation factor (VIF), or a condition index. Here we used correlation matrix and determined the independent variables.

Models -

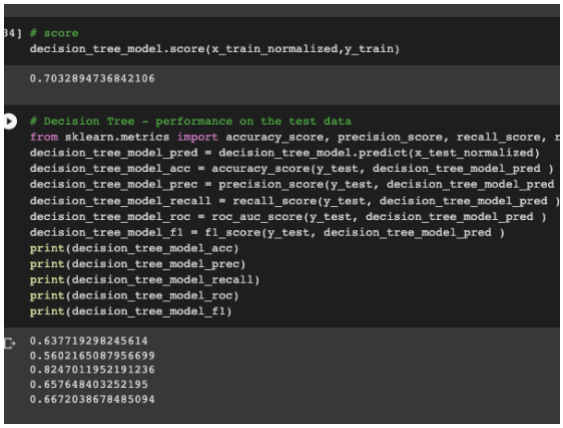
The algorithms implemented include decision tree, Logistic algorithm, and random forest.

We also trained the model with KNN and SVM but the accuracy was low. So we decided to use decision tree, Logistic algorithm, and random forest which got better results.

A decision tree is a supervised learning technique used for classification and regression tasks. It is a tree-like structure in which each branch represents a choice being made, each node represents a test on an attribute, and each leaf node represents a class label. The method works by building a decision tree from the training data and then utilizing it to generate predictions on unseen data.

We have trained the model with feature with normalized data of x_train and target feature of y_train.

The accuracy of the decision tree algorithm on training is 70.3% and on the testing dataset it was 63.7%.

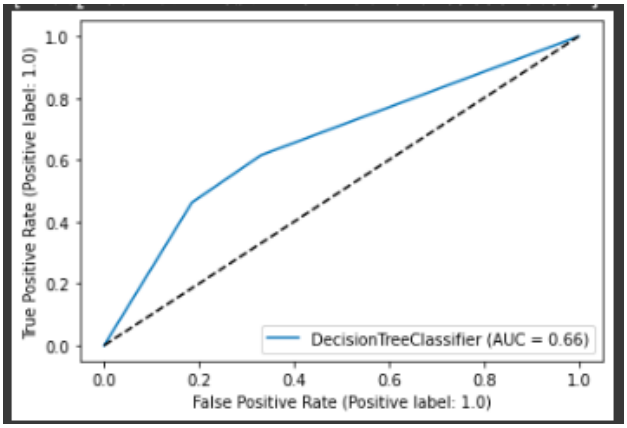


```
94] # score
    decision_tree_model.score(x_train_normalized,y_train)

    0.7032894736842106

    # Decision Tree - performance on the test data
    from sklearn.metrics import accuracy_score, precision_score, recall_score, r
    decision_tree_model_pred = decision_tree_model.predict(x_test_normalized)
    decision_tree_model_acc = accuracy_score(y_test, decision_tree_model_pred )
    decision_tree_model_prec = precision_score(y_test, decision_tree_model_pred
    decision_tree_model_recall = recall_score(y_test, decision_tree_model_pred )
    decision_tree_model_roc = roc_auc_score(y_test, decision_tree_model_pred )
    decision_tree_model_f1 = f1_score(y_test, decision_tree_model_pred )
    print(decision_tree_model_acc)
    print(decision_tree_model_prec)
    print(decision_tree_model_recall)
    print(decision_tree_model_roc)
    print(decision_tree_model_f1)

    0.637719298245614
    0.5602165087956699
    0.824701195219123
    0.657648403252195
    0.667203867848509
```



Accuracy on Test Data ROC Curve

Random Forest is a supervised learning technique that may be used for classification as well as regression applications. It is an ensemble learning approach that integrates numerous decision trees to get a more accurate and stable prediction.

We have trained the model with feature with normalized data of x_train and target feature of y_train.

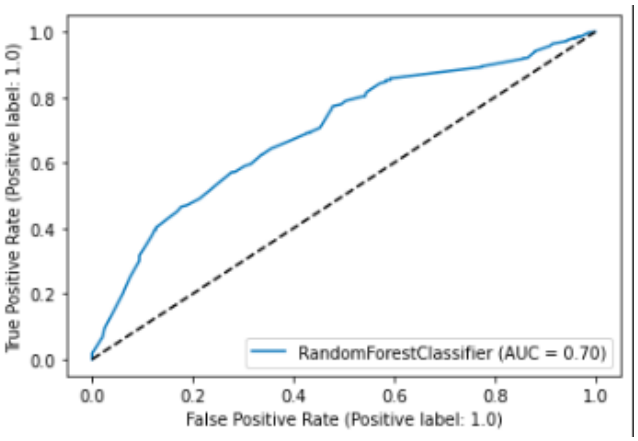
The accuracy of the Random Forest algorithm on training is 74.8% and on the testing dataset it was 72.3%.

```
random_forest_model.score(x_train_normalized,y_train)

<ipython-input-51-15a15f0089aa>:4: DataConversionWarning: A colu
  random_forest_model.fit(x_train_normalized,y_train)
0.7489035087719298

# Random Forest - performance on the test data
from sklearn.metrics import accuracy_score, precision_score, rec
random_forest_model_pred = random_forest_model.predict(x_test_no
random_forest_model_acc = accuracy_score(y_test, random_forest_m
random_forest_model_prec = precision_score(y_test, random_forest
random_forest_model_recall = recall_score(y_test, random_forest_
random_forest_model_roc = roc_auc_score(y_test, random_forest_mo
random_forest_model_f1 = f1_score(y_test, random_forest_model_pr
print(random_forest_model_acc)
print(random_forest_model_prec)
print(random_forest_model_recall)
print(random_forest_model_roc)
print(random_forest_model_f1)

0.7236842105263158
0.6643233743409490
0.7529880478087649
0.7268075035282069
0.7058823529411764
```



Accuracy on Test Data ROC Curve

Logistic Regression is a supervised machine learning technique used for classification problems. It is used when the target variable is a category variable (binary or multiclass). The goal of logistic regression is to identify the best fitting model to represent the connection between the dependent variable (the one we are attempting to predict/explain) and one or more independent variables (also known as predictors). Logistic regression calculates the likelihood that an observation belongs to a given category or class. It then chooses the class with the highest probability as the forecast output.

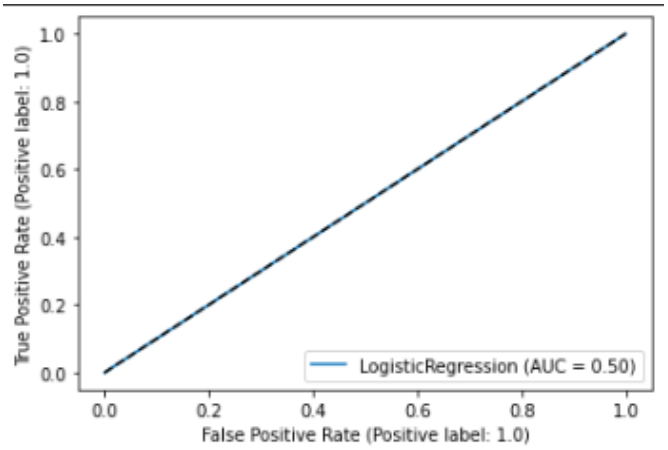
We have trained the model with feature with normalized data of x_train and target feature of y_train.

The accuracy of the Random Forest algorithm on training is 72.5% and on the testing dataset it was 704%.

```
[39] logistic_model.score(x_train_normalized,y_train)
Files
     0.725219298245614

[40] from sklearn.metrics import accuracy_score, precision_sco
     # Logistic regression - performance on the test data
     logistic_pred = logistic_model.predict(x_test_normalized)
     logistic_acc = accuracy_score(y_test, logistic_pred )
     logistic_prec = precision_score(y_test, logistic_pred )
     logistic_recall = recall_score(y_test, logistic_pred )
     logistic_roc = roc_auc_score(y_test, logistic_pred )
     logistic_f1 = f1_score(y_test, logistic_pred )
     print(logistic_acc)
     print(logistic_prec)
     print(logistic_recall)
     print(logistic_roc)
     print(logistic_f1)

     0.7043859649122807
     0.6248108925869894
     0.8227091633466136
     0.7169972149021469
     0.7102321582115219
```



Accuracy on Test Data ROC Curve

```
[55] evaluation_metrics
```

	Methods	Accuracy	Precision	Recall	F1-score	AUC score
0	Decision Tree	0.637719	0.560217	0.824701	0.667204	0.657648
1	SVM	0.718421	0.661319	0.739044	0.698024	0.720619
2	KNN	0.714912	0.665421	0.709163	0.686596	0.714300
3	Random Forest	0.723684	0.664323	0.752988	0.705882	0.726808
4	Logistic regression	0.704386	0.624811	0.822709	0.710232	0.716997

Accuracy, precision, recall, F1 score, AUC-ROC curve, confusion matrix, and other metrics can be used to evaluate the ML model/results. The accuracy statistic reflects the percentage of right predictions, while the precision, recall, and F1 score assess the model's ability to distinguish between positive and negative classifications. By comparing the true positive rate against the false positive rate, the ROC curve is used to evaluate the model's performance. The confusion matrix is used to assess the performance of the model by visualizing the number of accurate and wrong predictions. Finally, sensitivity, specificity, and the Matthews correlation coefficient may be utilized to evaluate the ML model.

Random Forest Machine Learning Model is best model with high accuracy.

Results

Cross-validation is a machine learning approach for model selection and parameter adjustment. It is a method for evaluating the performance of a machine learning model on a given dataset by dividing the data into different subsets, training the model on one subset, and testing it on the other. To obtain an accurate measurement of the model's performance, the process is performed several times. Cross-validation is an effective method for determining the best model and fine-tuning the parameters of a machine learning algorithm.

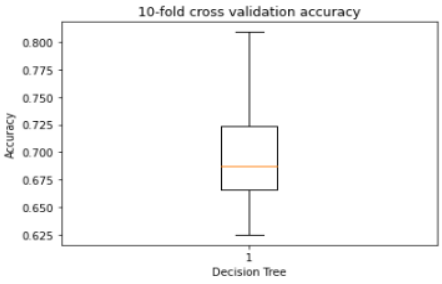
The algorithms implemented were tested on the testing datasets and the results obtained were as follows:

Decision Tree –

The following are the results of the cross validations and for decision tree

```
[56] # Cross Validation for Decision Tree
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.model_selection import cross_val_score
     CV_scores_decision_tree_model = cross_val_score(estimator = decision_tree_model,
     print("CV_scores: ", CV_scores_decision_tree_model)

     CV_scores:  [0.66885965 0.6995614  0.66008772 0.74780702 0.70614035 0.65350877
      0.73464912 0.70175439 0.67982456 0.67763158]
```

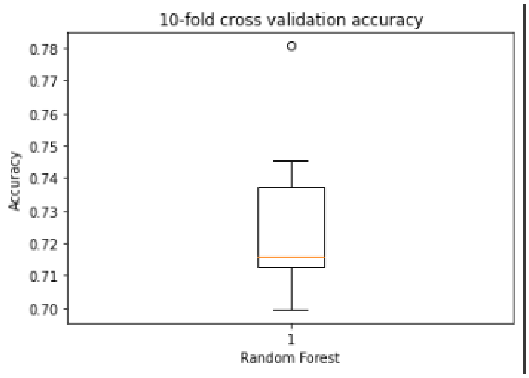


Random Forest –

The following are the results of the cross validations and for Random Forest

```
# Cross validation for Random Forest
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
model1 = random_forest_model
CV_scores_random_forest_model = cross_val_score(estimator = model1, X = X_train, y = y_train, cv = 5, scoring = 'accuracy')
print("CV_scores: ", CV_scores_random_forest_model)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/model_selection/_validation.py:680: DataConversionWarning: A column-vector y was
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.8/dist-packages/sklearn/model_selection/_validation.py:680: DataConversionWarning: A column-vector y was
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.8/dist-packages/sklearn/model_selection/_validation.py:680: DataConversionWarning: A column-vector y was
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.8/dist-packages/sklearn/model_selection/_validation.py:680: DataConversionWarning: A column-vector y was
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.8/dist-packages/sklearn/model_selection/_validation.py:680: DataConversionWarning: A column-vector y was
  estimator.fit(X_train, y_train, **fit_params)
CV_scores:  [0.72039474 0.73026316 0.73026316 0.72807018 0.72587719]
```



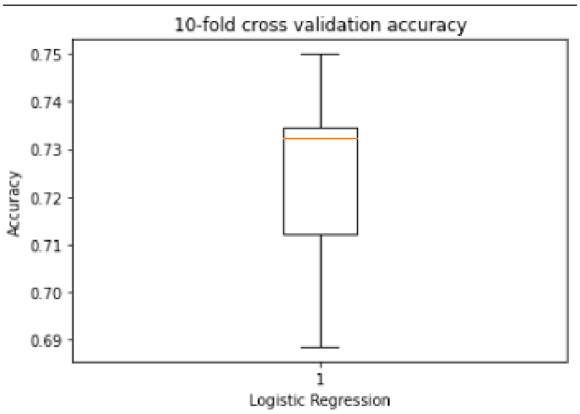
Logistic Regression –

The following are the results of the cross validations and for Logistic Regression

```
# Cross validation for Logictic regression
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
CV_scores_lr = cross_val_score(estimator = logistic_model, X = X_train, y = y_train, cv = 10, scoring = 'accuracy')
print("CV_scores: ", CV_scores_lr)
```

```
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:993: DataConversi
  y = column_or_1d(y, warn=True)
CV_scores:  [0.73245614 0.73464912 0.68859649 0.75       0.73464912 0.73245614
 0.74122807 0.73026316 0.70175439 0.70614035]
/usr/local/lib/python3.8/dist-packages/sklearn/linear_model/_logistic.py:814: Conver
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max iter) or scale the data as shown in:
```



Box Plot

ROC curves are graphical representations of the performance of a classification model. The ROC curve's x-axis shows the false positive rate, while the y-axis represents the true positive rate. ROC curves are used to assess the performance of different classifiers and to identify the best cut-off value for a particular model. We used it on the all models that are classified.

After Evaluating all models, Random Forest Machine Learning Algorithm has given best results.

Contributions

Member 1:

- Researching and understanding the problem statement
- Identifying the type of Machine Learning algorithm to be used
- Determining the performance metrics for the model
- Preparing the data for building the model
- Developing the model using appropriate ML algorithm
- Project report

Member 2:

- Collecting and exploring the data for the project
- Cleaning, normalizing and pre-processing the data
- Feature engineering and selection
- Implementing the model

Member 3:

- Analysing the model performance and accuracy
- Conducting model validation and hyperparameter tuning
- Generating insights and business value from the model results
- Deploying the model
- Monitoring and maintaining the model

Learnings, Further Discussions and Implementations

This introduction to machine learning course has offered an overview of the principles of machine learning and its applications. We have learnt the fundamental principles and techniques of supervised and unsupervised learning, as well as how to apply them to real-world issues. Furthermore, we have obtained programming and data analysis knowledge using Python, as well as familiarity with numerous machine learning methods such as linear regression, logistic regression, decision trees, support vector machines, k-means clustering, and neural networks. Finally, we had the opportunity to put their newly gained knowledge to use by building machine learning models to fix actual issues. Further we can improve the model, add more features.