Drug Classification

Merve Rana Kızıl
150119825

Sueda Bilen
150117044

Elif Gülay
150119732

***Abstract*—The purpose of the project is to try to find the most suitable drug for the patients. A dataset will be used which includes data on drug classification based on the diagnosis of the patient and general characteristics, such as age, sex. Different machine learning algorithms that can predict the type of the drug suitable for the patient will be applied and the results of each will be compared.**

***Index Terms*—drug, patient, machine learning, classification, k-NN, naive bayes, random forest, support vector machines, decision tree**

I. Introduction

Drugs are the substances that can be used for preventing pain or symtoms, treatment of an illness, or a condition unexpected. Even in daily life people using drugs to relieve pain, support their immune system etc.

In this project, we will try to predict the correct drugs for the patients by analysing their gender, age, cholesterol, blood pressure levels, and sodium to potassium values in the dataset.

Fig. 1. Drugs

II. Tasks

A project subject will be defined and a literature search will be made about this subject. After that, the features of the dataset will be explored and exploratory data analysis will be made by using various plots. Dataset will be prepared and five different algorithms will be implemented to predict the the drug types for the patients. Obtained results of the algorithms will be compared and visualization will be made.

Planned workflow of the project can be seen below.

A. Workflow

1) Define the project subject
2) Initial literature search
3) Dataset exploration
4) Exploratory data analysis
5) Dataset preparation
6) Implementation of different algorithms
   a) k-NN
   b) Naive Bayes
   c) Random Forest
   d) Support Vector Machines
   e) Decision Tree
7) Visualization and the comparison of the algorithms
8) Conclusion

By the time until midterm report: Analysis and the exploration of the dataset will be made and two different algorithms will be implemented which are k-NN and Naive Bayes. Dataset will be visualized by plots and the results of the algorithms will be compared using Python.

After midterm report: The rest of the algorithms will be implemented which are Random Forest, Support Vector Machines and Decision Tree. Plots will be used to compare these algorithms.

III. Dataset and Features

The dataset that will be used in this project is obtained from https://www.kaggle.com/datasets/prathamtripathi/drug-classification. It contains information about patients' age, sex, blood pressure levels, cholesterol levels and sodium to potassium ration in blood. It contains information about 200 patients, i.e. there are 200 samples in the dataset.

Representative dataset is given in TABLE I and the descriptions of the variables are given in TABLE II.

Age	Sex	BP	Cholesterol	Na-to-K	Drug
23	F	HIGH	HIGH	25.355	DrugY
47	M	LOW	HIGH	13.093	DrugC
28	F	NORMAL	HIGH	7.798	DrugX
67	M	NORMAL	NORMAL	10.898	DrugX
36	F	HIGH	HIGH	11.198	DrugA

TABLE I
Representative dataset.

Variable Name	Description
Age	Age of the patient
Sex	Gender of the patient (male or female)
BP	Levels of blood pressure (high, normal, or low)
Cholesterol	Levels of cholesterol (high or normal)
Na-to-K	Sodium to potassium ratio in blood
Drug	Type of the drug

TABLE II
Description of the variables

IV. EXAMINATION AND VISUALIZATION OF DATASET

```
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Age          200 non-null    int64
 1   Sex          200 non-null    object
 2   BP           200 non-null    object
 3   Cholesterol  200 non-null    object
 4   Na_to_K      200 non-null    float64
 5   Drug         200 non-null    object
```

Fig. 2. Summary of Dataset Variables

There are 6 variables in the dataset; Age, Sex, BP, Cholesterol, Na-To-K and Drug. There are 200 examples in dataset, which is a bit small. From the results above, there are no missing or null value in this dataset .

Feature 1; Age. Age distribution mostly concentrates on 0.30 which means 30 as shared below.

```
Age skewness:  0.03030835703000607
```

Fig. 3. Age skewness

Age interval is starting from 20 goes up to 70+ and there is 2 peaks on graph.

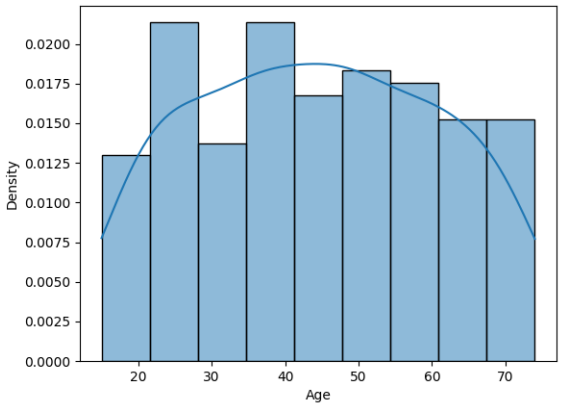


Fig. 4. Age skewness

Feature 2; Sex. Sex is distributed almost equivalently in the dataset as in the Fig.5.

```
M    104
F     96
Name: Sex, dtype: int64
```

Fig. 5. Sex

In Fig.6 below, we can see the distribution visualized.

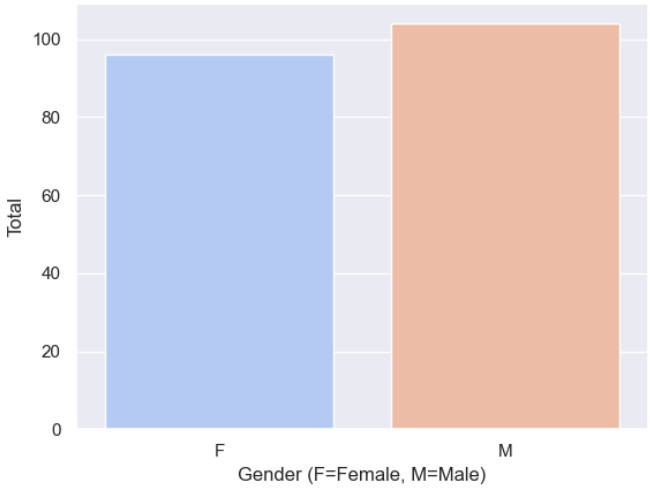


Fig. 6. Gender Distribution Graph

Feature 3; Drug. There are 5 types of drugs, and most of the data belong to Drug Y. Then Drug X follows, however for Drug A, B, and C least ones among them.

```
None
DrugY    91
drugX    54
drugA    23
drugC    16
drugB    16
Name: Drug, dtype: int64
```

Fig. 7. Drug

As visualized, we can see data density on DrugY and DrugX.

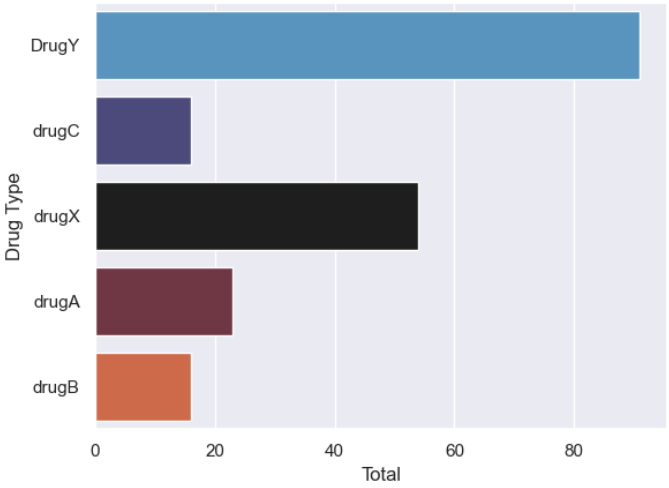


Fig. 8. Drug

Feature 4; BP (Blood Pressure). Blood pressure can be High, Low or Normal, and distributed equally in the data set.

```
HIGH      77
LOW       64
NORMAL    59
Name: BP, dtype: int64
```

Fig. 9. BP

It can be observed that Blood Pressure values on the data set are equally distributed when Fig.10 is considered.

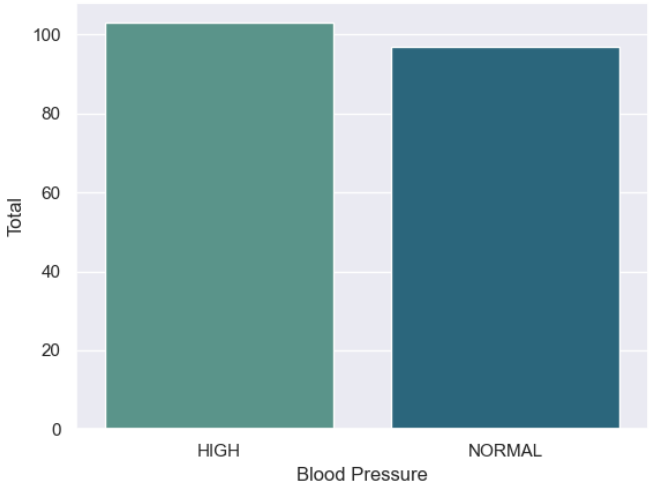


Fig. 10. BP

Feature 5; Cholesterol. Cholesterol can be High or Normal, and distributed equally in the data set.

```
HIGH      103
NORMAL     97
Name: Cholesterol, dtype: int64
```

Fig. 11. Cholesterol

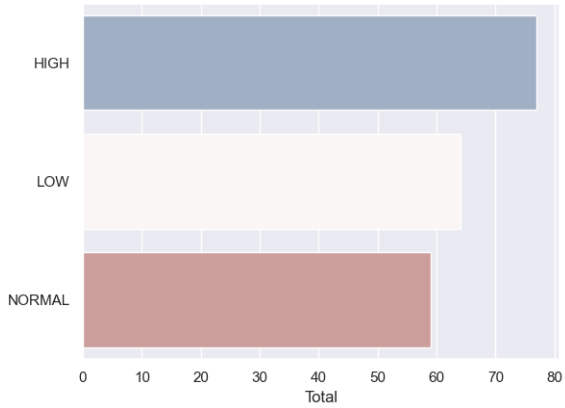


Fig. 12. Cholesterol

Feature 6; Na-To-K (Na to Potassium Ration). Na-To-K is the result of sodium divided by potassium and according to that ratio, some of the criteria such as dietary quality or intake, and risk of diseases.

```
Na to K skewness:  1.039341186028881
```

Fig. 13. Na_To_K

In the plot, we can see Na-To-K versus Age and observed as equally distributed in the data set. Most of the data have low Na-To-K values.

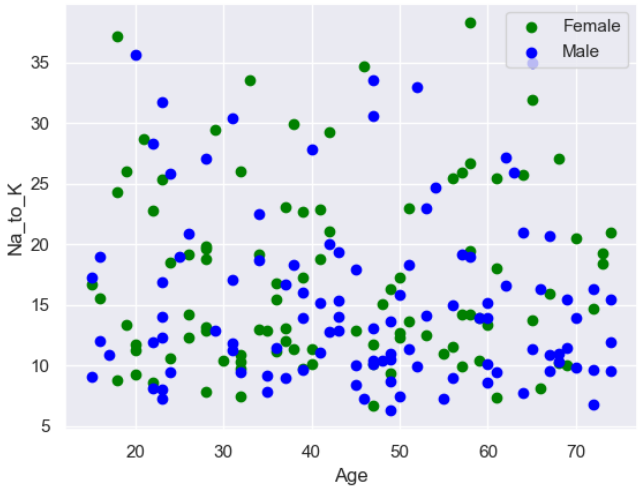


Fig. 14. Na_To_K Plot

Before starting the training and testing steps the data set is split into two parts. 70% of the data set is used for training, and the rest, 30%, is used for testing.

V. METHODS

A. k-NN

k-Nearest Neighbor (k-NN) is one of the machine learning algorithms used for classification. The aim of the algorithm is to find a class for a new data point based on the similarity between available data. The k-NN algorithm is based on the logic of assigning the new data point to the class of its nearest neighbor.

B. Naive Bayes

Naive Bayes is a classification algorithm based on Bayes' theorem. Naive Bayes provides a mechanism for using the information in sample data to estimate the posterior probability P(y—x) of each class y given an object x. Once we have such estimates we can use them for classification [1].

C. Random Forest

Random Forest is one of the supervised classification algorithms. Random forest is an ensemble classifier, which constructs a group of independent and non-identical decision trees based on the idea of randomization [2]. It is also can be defined as a Decision Tree-Based Classifier that chooses the best classification tree as the final classifier's classification of the algorithm via voting [3].

D. Support Vector Machines

Support Vector Machines are a combination of supervised learning methods gathered together in last decades that uses linear and non-linear classification. Support Vector Machine aims to connect training data to the points in the hyperplane with maximizing the space between two different categories. When a data came it will be predicted according to the side it falls into.

E. Decision Tree

Decision Tree is a Tree - Structured approach that parts data into root and leaf nodes to classify. With that method, decision tree can be learned from the data taken. At the leaf nodes, decision (class of the taken data) can be made [4].

VI. IMPLEMENTATION

k-NN and Naive Bayes algorithms are implemented. The implementation steps for both are given below:

1) Examination of the data
2) Data pre-processing
3) Oversampling
4) Training and testing the model

Examination of the data

This part is analysed and completed in section IV. Examination and Visualization of Data Set.

Data pre-processing Data binning is applied and the age and chemical ratio features in our data set are divided into 7 categories for feature age, and 4 categories for feature chemical.

Data pre-processing is used to prepare the data set and enable feature engineering. Categorical encoding is one of the methods of feature engineering. This method is used to encode categorical features into numerical values so the data can be understood simply by the machine learning models. One hot encoding is a method of categorical encoding. In one hot encoding, categorical values are converted into binary values, i.e. 0s and 1s.

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	DrugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	DrugY

Fig. 15. Representative data set before pre-processing

	Sex_F	Sex_M	BP_HIGH	BP_LOW	BP_NORMAL	Cholesterol_HIGH	...
131	0	1	0	1	0	0	...
96	1	0	0	1	0	1	...
181	1	0	0	0	1	1	...
19	1	0	1	0	0	0	...
153	1	0	0	1	0	0	...

Fig. 16. Representative data set after pre-processing

As can be seen in Figure 15, one-hot coding is implemented and features are represented as 0s and 1s.

Oversampling

Bias in the training data set can affect the machine learning models, oversampling is used to overcome this problem. One of the oversampling methods is SMOTE: Synthetic Minority Over-sampling Technique. This technique is used to oversample the minority class.

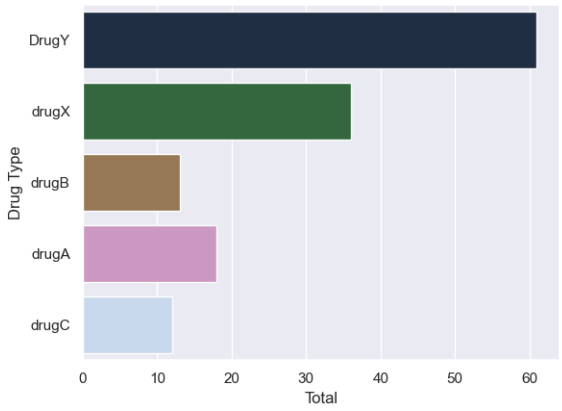


Fig. 17. Numerical representation of the drug types before using SMOTE

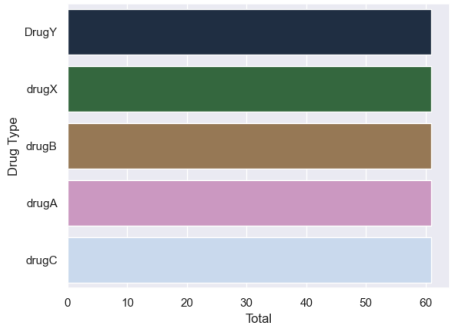


Fig. 18. Numerical representation of the drug types after using SMOTE

If Figure 16 and Figure 17 are compared, it can be seen that the minority drug types are oversampled according to the count of the major drug type.

Training and testing the model

Machine learning models should first be trained using a certain size of data and then tested with data different from the data used in the training phase. In this way, if these previously unseen data can be predicted correctly, it is understood that the training phase is successful. For this purpose, we divided our data set into two, 70% to be used in the training phase and 30% to be used in the testing phase.

VII. RESULTS

k-NN, Naive Bayes, Random Forest, Support Vector Machines, and Decision Tree algorithms are implemented.

k-NN

k-NN

	precision	recall	f1-score	support
DrugY	0.86	0.63	0.73	30
drugA	0.50	0.80	0.62	5
drugB	0.33	0.33	0.33	3
drugC	0.57	1.00	0.73	4
drugX	0.80	0.89	0.84	18
accuracy			0.73	60
macro avg	0.61	0.73	0.65	60
weighted avg	0.77	0.73	0.73	60

K Neighbours accuracy is: 73.33%

Fig. 19. Classification report of the k-NN algorithm

Precision gives information about the accuracy of positive predictions. In Figure 19, among other drug types, DrugY has the highest precision value, which is 95%. Recall shows the percentage of finding all positive cases. Finally, the f1-score indicates what percentage of positive predictions are correct.

The accuracy rate of k-NN is found as 73.33%.

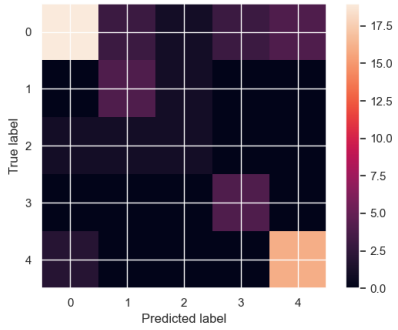


Fig. 20. Confusion matrix of the k-NN algorithm

Confusion matrices of each algorithm are plotted to visualize the precision value of each drug type. In Figure 20 confusion matrix's diagonal has light colors. This indicates that the majority of the predictions are correct. DrugY is shown with the value of 0 in the matrix, which has the lightest color on the matrix since it has the highest precision among the other drug types.

Naive Bayes

Naive Bayes

	precision	recall	f1-score	support
DrugY	0.84	0.70	0.76	30
drugA	0.71	1.00	0.83	5
drugB	0.75	1.00	0.86	3
drugC	0.67	1.00	0.80	4
drugX	0.78	0.78	0.78	18
accuracy			0.78	60
macro avg	0.75	0.90	0.81	60
weighted avg	0.79	0.78	0.78	60

Naive Bayes accuracy is: 78.33%

Fig. 21. Classification report of the Naive Bayes algorithm

In Figure 21, among other drug types, DrugY has the highest precision value, which is 95%. The accuracy rate of the Naive Bayes is found as 78.33%.

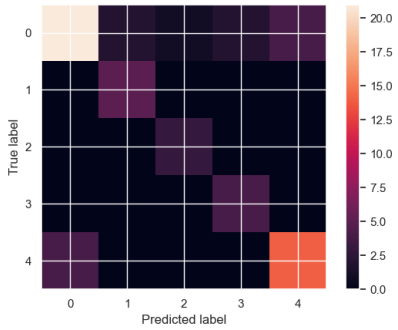


Fig. 22. Confusion matrix of the Naive Bayes algorithm

In Figure 22 confusion matrix's diagonal has light colors. This indicates that the majority of the predictions are correct.

DrugY is shown with the value of 0 in the matrix, which has the lightest color on the matrix since it has the highest precision among the other drug types.

Random Forest

Random Forest

	precision	recall	f1-score	support
DrugY	1.00	0.67	0.80	30
drugA	0.62	1.00	0.77	5
drugB	0.75	1.00	0.86	3
drugC	0.67	1.00	0.80	4
drugX	0.82	1.00	0.90	18
accuracy			0.83	60
macro avg	0.77	0.93	0.83	60
weighted avg	0.88	0.83	0.83	60

Random Forest accuracy is: 83.33%

Fig. 23. Classification report of the Random Forest algorithm

In Figure 23, among other drug types, DrugY has the highest precision value, which is 100%. The accuracy rate of the Naive Bayes is found as 83.33%.

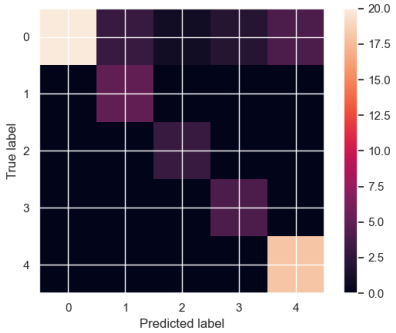


Fig. 24. Confusion matrix of the Random Forest algorithm

In Figure 24 confusion matrix's diagonal has light colors. This indicates that the majority of the predictions are correct. DrugY is shown with the value of 0 in the matrix, which has the lightest color on the matrix since it has the highest precision among the other drug types with the value of 100%.

Support Vector Machines

Support Vector Machines

	precision	recall	f1-score	support
DrugY	0.95	0.70	0.81	30
drugA	0.67	0.80	0.73	5
drugB	0.75	1.00	0.86	3
drugC	0.67	1.00	0.80	4
drugX	0.82	1.00	0.90	18
accuracy			0.83	60
macro avg	0.77	0.90	0.82	60
weighted avg	0.86	0.83	0.83	60

SVM accuracy is: 83.33%

Fig. 25. Classification report of the SVM algorithm

In Figure 25, among other drug types, DrugY has the highest precision value, which is 95%. The accuracy rate of the Naive Bayes is found as 83.33%.

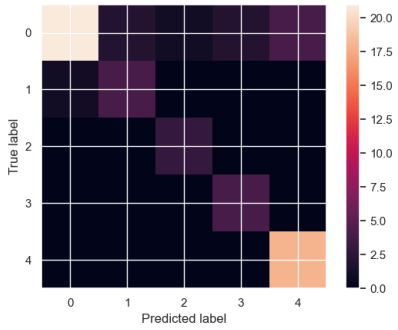


Fig. 26. Confusion matrix of the SVM algorithm

In Figure 26 confusion matrix's diagonal has light colors. This indicates that the majority of the predictions are correct. DrugY is shown with the value of 0 in the matrix, which has the lightest color on the matrix since it has the highest precision among the other drug types. After DrugY, drug type X follows with the second highest precision value, which is shown with a darker color in Figure 26.

Decision Tree

Decision Tree

	precision	recall	f1-score	support
DrugY	1.00	0.63	0.78	30
drugA	0.56	1.00	0.71	5
drugB	0.75	1.00	0.86	3
drugC	0.67	1.00	0.80	4
drugX	0.82	1.00	0.90	18
accuracy			0.82	60
macro avg	0.76	0.93	0.81	60
weighted avg	0.87	0.82	0.81	60

Decision Tree accuracy is: 81.67%

Fig. 27. Classification report of the Decision Tree algorithm

In Figure 27, among other drug types, DrugY has the highest precision value, which is 100%. The accuracy rate of the Naive Bayes is found as 81.67%.

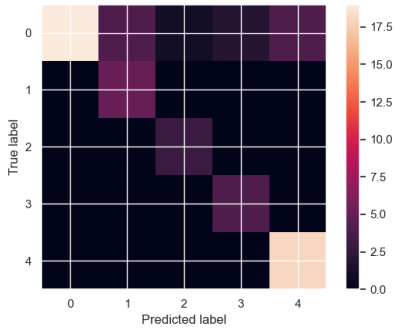
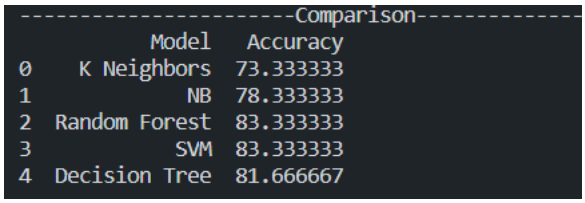


Fig. 28. Confusion matrix of the Decision Tree algorithm

In Figure 28 confusion matrix's diagonal has light colors. This indicates that the majority of the predictions are correct. DrugY is shown with the value of 0 in the matrix, which has the lightest color on the matrix since it has the highest precision among the other drug types.

Comparison of the Algorithms



----------Comparison----------

	Model	Accuracy
0	K Neighbors	73.333333
1	NB	78.333333
2	Random Forest	83.333333
3	SVM	83.333333
4	Decision Tree	81.666667

Fig. 29. Accuracy percentages of the algorithms

The accuracy percentages of the algorithms can be seen in Figure 29. The Random Forest and the Support Vector Machines have the highest accuracy rate with 83.33%. The k-NN algorithm has the lowest accuracy rate with 73.33%.

VIII. CONCLUSION

Examination and visualization of the data set are performed. k-NN, Naive Bayes, Random Forest, Support Vector Machines, and Decision Tree algorithms are implemented. The accuracy rates of each algorithm are calculated. A classification report and a confusion matrix for each algorithm are printed. Finally, a comparison of these algorithms is made. It is seen that the accuracy rates of the algorithms have low values which may be caused by the small sample size of the data set. To obtain better accuracy rates, the number of samples in the data set may be increased.

IX. APPENDIX

Source codes of the project are given in this section.

A. *main.py*

```
import pandas as pd
from Dataset import *
from Algorithm import *

def main():
    df_drug = pd.read_csv("drug200.csv")
    print("Dataset head: ")
    print(df_drug.head())
    # check null in dataset
    print("Dataset info:")
    print(df_drug.info())

    dataset = Dataset(df_drug)
    dataset.df_drug = df_drug

    dataset.
    explore_categorical_variables()
    dataset.explore_numerical_variables()
    dataset.exploratory_data_analysis()
    dataset.data_bining()
    X_train, X_test, y_train, y_test =
    dataset.split_dataset()
    X_train, X_test = dataset.
    feature_engineering(X_train, X_test)
    dataset.check_number_of_methods
    (y_train)
    X_train, y_train =
    dataset.smote(X_train, y_train)
    dataset.check_number_of_methods
    (y_train)
    print(X_train.head())

    algorithm = Algorithm
    (X_train, X_test, y_train, y_test)
    algorithm.knn()
    algorithm.naive_bayes()
    algorithm.random_forest()
    algorithm.SVM()
    algorithm.decision_tree()
    algorithm.algorithm_comparison()

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

B. *Dataset.py*

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection
import train_test_split
from imblearn.over_sampling
import SMOTE
from collections import Counter


class Dataset:
    def __init__(self, df_drug):
        self.df_drug = df_drug

    def data_bining(self):
        # divide age into 7 categories
        bin_age = [0, 19, 29, 39, 49, 59,
        69, 80]
        category_age = ['<20s', '20s',
        '30s',
        '40s', '50s', '60s', '>60s']
        self.df_drug['Age_binned'] =
        pd.cut(self.df_drug['Age'],
        bins=bin_age, labels=category_age)
        self.df_drug = self.df_drug.drop
        (['Age'], axis = 1)
```

```
        # divide chemical ratio
        # into 4 categories
        bin_NatoK = [0, 9, 19, 29, 50]
        category_NatoK = ['<10', '10-20',
        '20-30', '>30']
        self.df_drug['Na_to_K_binned'] =
        pd.cut(self.df_drug['Na_to_K'],
        bins=bin_NatoK,
        labels=category_NatoK)
        self.df_drug =
        self.df_drug.drop(['Na_to_K'],
        axis = 1)


    def explore_categorical_variables
    (self):
        print(
        self.df_drug.Drug.value_counts())
        print(
        self.df_drug.Sex.value_counts())
        print(
        self.df_drug.BP.value_counts())
        print(
        self.df_drug.Cholesterol
        .value_count())


    def explore_numerical_variables(self):
        skewAge = self.df_drug.Age.skew
        (axis = 0, skipna = True)
        print('Age skewness: ', skewAge)
        sns.histplot(self.df_drug['Age'],
        kde=True, stat="density")
        plt.show()

        skewNatoK = self.df_drug.Na_to_K
        .skew(axis = 0, skipna = True)
        print('Na to K skewness: ',
        skewNatoK)
        sns.histplot(self.df_drug
        ['Na_to_K'], kde=True,stat=
        "density")
        plt.show()


    def exploratory_data_analysis(self):
        # drug type distribution
        sns.set_theme(style="darkgrid")
        sns.countplot(y = "Drug", data =
        self.df_drug, palette = "icefire")
        plt.ylabel('Drug Type')
        plt.xlabel('Total')
        plt.show()

        # gender distribution
        sns.set_theme(style="darkgrid")
        sns.countplot(x = "Sex", data =
        self.df_drug, palette =
        "coolwarm")
        plt.xlabel('Gender(F=Female,
        M=Male)')
        plt.ylabel('Total')
        plt.show()

        # blood pressure distribution
        sns.set_theme(style="darkgrid")
        sns.countplot(y="BP",
        data = self.df_drug, palette=
        "vlag")
        plt.ylabel('Blood Pressure')
        plt.xlabel('Total')
        plt.show()

        # cholesterol distribution
        sns.set_theme(style="darkgrid")
        sns.countplot(x = "Cholesterol",
        data = self.df_drug,
        palette = "crest")
        plt.xlabel('Blood Pressure')
        plt.ylabel('Total')
        plt.show()

        # gender distribution based on
        # drug type
        pd.crosstab(self.df_drug.Sex,
        self.df_drug.Drug).plot(kind="bar",
        figsize=(12,5),color=['#9BC2B2',
        '#C5D6BA','#F2E9D3','#F6C8B6',
        '#CA9CAC'])
        plt.title('Gender distribution
        based on Drug type')
        plt.xlabel('Gender')
        plt.xticks(rotation = 0)
        plt.ylabel('Frequency')
        plt.show()

        # blood pressure distribution
        # based on cholesterol
        pd.crosstab(self.df_drug.BP,
        self.df_drug.Cholesterol).plot(
        kind="bar",figsize=(15,6),
        color=['#FFCBB5','#F28589'])
        plt.title('Blood Pressure
        distribution based on
        Cholesterol')
        plt.xlabel('Blood Pressure')
        plt.xticks(rotation=0)
        plt.ylabel('Frequency')
        plt.show()

        # sodium to potassium
        # distribution based on gender
```

```
        # and age
        plt.scatter(x = self.df_drug.Age
        [self.df_drug.Sex==’F’],
        y = self.df_drug.Na_to_K[
        (self.df_drug.Sex==’F’)],
        c="Green")
        plt.scatter(x =
        self.df_drug.Age[
        self.df_drug.Sex==’M’],
        y = self.df_drug.Na_to_K[
        (self.df_drug.Sex==’M’)],
        c="Blue")
        plt.legend(["Female", "Male"])
        plt.xlabel("Age")
        plt.ylabel("Na_to_K")
        plt.show()


    def split_dataset(self):
        X = self.df_drug.drop(["Drug"],
        axis=1)
        y = self.df_drug["Drug"]
        X_train, X_test, y_train, y_test
        = train_test_split(X, y,
        test_size = 0.3,
        random_state = 0)
        return X_train, X_test, y_train,
        y_test


    def feature_engineering(self, X_train,
    X_test):
        X_train = pd.get_dummies(X_train)
        X_test = pd.get_dummies(X_test)
        return X_train, X_test


    def smote(self, X_train, y_train):
        SMOTE().fit_resample
        (X_train, y_train)
        return X_train, y_train


    def check_number_of_methods(self,
    y_train):
        # Check the number of records
        # after oversampling
        print(sorted
        (Counter(y_train).items()))
        sns.set_theme(style="darkgrid")
        sns.countplot(y=y_train, data =
        self.df_drug, palette="cubehelix")
        plt.ylabel(’Drug Type’)
        plt.xlabel(’Total’)
        plt.show()
```

C. Algorithm.py

```
from sklearn.naive_bayes import
CategoricalNB
from sklearn.neighbors import
KNeighborsClassifier
from sklearn.metrics import
confusion_matrix
from sklearn.metrics import
classification_report
from sklearn.neighbors import
KNeighborsClassifier
from sklearn.naive_bayes import
CategoricalNB
from sklearn.svm import SVC
from sklearn.ensemble import
RandomForestClassifier
from sklearn.tree import
DecisionTreeClassifier
from sklearn.metrics import
accuracy_score
import matplotlib.pyplot as plt
from Dataset import *
from Algorithm import *

class Algorithm:
    def __init__(self, X_train, X_test,
    y_train, y_test):
        self.X_train, self.X_test,
        self.y_train, self.y_test =
        X_train, X_test, y_train,
        y_test
        # accuracies of each algorithm
        self.knn_acc = 0
        self.nb_acc = 0
        self.rf_acc = 0
        self.svm_acc = 0
        self.dt_acc = 0


    def knn(self):
        KNclassifier =
        KNeighborsClassifier(n_neighbors
        =20)
        KNclassifier.fit(self.X_train,
        self.y_train)
        y_pred = KNclassifier.predict
        (self.X_test)
        print()
        print(classification_report(
        self.y_test, y_pred))
        self.knn_acc = accuracy_score(
        y_pred, self.y_test)
        print(’K Neighbours accuracy is:
```

```
        {:.2f}%'.format(self.knn_acc*100))
        self.plot_confusion_matrix(
        self.y_test, y_pred)


    def naive_bayes(self):
        NBclassifier = CategoricalNB()
        NBclassifier.fit(self.X_train,
        self.y_train)
        y_pred =
        NBclassifier.predict(self.X_test)
        print()
        print(classification_report(
        self.y_test, y_pred))
        self.nb_acc = accuracy_score(
        y_pred, self.y_test)
        print('Naive Bayes accuracy is:
        {:.2f}%'.format(self.nb_acc*100))
        self.plot_confusion_matrix(
        self.y_test, y_pred)


    def random_forest(self):
        RFclassifier =
        RandomForestClassifier(
        max_leaf_nodes = 30)
        RFclassifier.fit(self.X_train,
        self.y_train)
        y_pred =
        RFclassifier.predict(self.X_test)
        print(classification_report(
        self.y_test, y_pred))
        self.rf_acc =
        accuracy_score(y_pred, self.y_test)
        print('Random Forest accuracy is:
        {:.2f}%'.format(self.rf_acc*100))
        self.plot_confusion_matrix(
        self.y_test, y_pred)


    def SVM(self):
        SVCclassifier = SVC(kernel=
        'linear', max_iter=251)
        SVCclassifier.fit(self.X_train,
        self.y_train)
        y_pred =
        SVCclassifier.predict(self.X_test)
        print(classification_report(
        self.y_test, y_pred))
        self.svm_acc =
        accuracy_score(y_pred,
        self.y_test)
        print('SVM accuracy is:
        {:.2f}%'.format(self.svm_acc*100))
        self.plot_confusion_matrix(
        self.y_test, y_pred)
```

```
    def decision_tree(self):
        DTclassifier =
        DecisionTreeClassifier(
        max_leaf_nodes=20)
        DTclassifier.fit(self.X_train,
        self.y_train)
        y_pred =
        DTclassifier.predict(self.X_test)
        print(classification_report(
        self.y_test, y_pred))
        self.dt_acc =
        accuracy_score(y_pred,
        self.y_test)
        print('Decision Tree accuracy is:
        {:.2f}%'.format(self.dt_acc*100))
        self.plot_confusion_matrix(
        self.y_test, y_pred)

    def algorithm_comparison(self):
        compare = pd.DataFrame({'Model':
        ['K Neighbors', 'NB', 'Random
        Forest', 'SVM', 'Decision Tree'],
        'Accuracy':
        [self.knn_acc*100,
        self.nb_acc*100,
        self.rf_acc*100,
        self.svm_acc*100,
        self.dt_acc*100]})
        compare.sort_values(by='Accuracy',
        ascending=False)
        print(compare)

    def plot_confusion_matrix(self,
    y_test, y_pred):
        conf_matrix =
        confusion_matrix(y_test, y_pred)
        plt.imshow(conf_matrix)
        plt.xlabel('Predicted label')
        plt.ylabel('True label')
        plt.colorbar()
        plt.show()
```

REFERENCES

[1] G. I. Webb, "Naïve Bayes," in Encyclopedia of Machine Learning and Data Mining, Springer US, 2016, pp. 1–2.

[2] Machine Learning Classification Based on Random Forest Algorithm: A Review, Mahdi Abdulkareem N, Mohsin Abdulazeez A, 2021

[3] Research on machine learning framework based on random forest algorithm, Ren Q, Cheng H, Han H, AIP Conference Proceedings, 2017

[4] https://www.cs.cmu.edu/ bhiksha/courses/10-601/decisiontrees/