A MAJOR PROJECT REPORT ON

SOFTWARE QUALITY PREDICTION USING MACHINE LEARNING

Submitted by

R.Sreelatha - R170553

Md.Shaistha Afreen - R170552

Submitted to

Department of CSE

IIIT RK Valley

Idupulapaya,Vempalli,YSR kadapa

Andhra Pradesh , India PIN 516330

Under the guidance of

N.Satyanandaram

Head of the department

as part of

Partial fulfillment of the degree of Bachelor of Technology in Computer Science and Engineering

CERTIFICATE

This is to certify that the project work entitled "**Software Quality prediction**" Submitted by **R.Sreelatha[R170553]** , **Md.Shaistha Afreen[R170552]** in partial Fulfillment of the requirements for award of Bachelor of Technology in computer Science and Engineering is a bonafide work carried out by them under my supervision and guidance.

The report has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Mr N.Satyanandaram,	Mr N.Satyanandaram,
Project Internal Guide,	Head of the department,
Computer Science and Engineering,	Computer Science andEngineering,
R.K Valley, RGUKT.	R.K.Valley , RGUKT.

Acknowledgement

I would like to express my sincere gratitude to Mr N. Satyanand ram sir, my project internal guide for valuable suggestions and keen interest throughout the progress of my course of research .

I am grateful to Mr N . Satyanand ram sir , HOD CSE, for providing excellent computing facilities and a congenial atmosphere for progressing with my project

My sincere thanks to all the members who helped me directly and indirectly in the completion of project work .I express my profound gratitude to all our friends and family members for their encouragement.

Table of Contents

Abstract

Software quality estimation is an activity needed at various stages of software development . It may be used for planning the project`s quality assurance practices and for benchmarking. In earlier previous studies , two methods (Multiple Criteria Linear Programming and Multiple Criteria Quadratic Programming) for estimating the quality of software had been used . Also, C5.0, SVM and Neural network were experimented with for quality estimation . These studies have relatively low accuracy.

In this study, we aimed to improve estimation accuracy by using relevant features of a large dataset . We used a feature selection method and correlation matrix for reaching higher accuracies.

In addition , we have experimented with recent methods shown to be successful for other prediction tasks . Machine learning algorithms such as XGBoost , Random Forest , Decision Tree, Logistic Regression and Naïve Bayes are applied to the data to predict the software quality and reveal the relation between the quality and development attributes..

The experimental results show that the quality level of software can be well estimated by machine learning algorithms.

Introduction

Software applications may contain defects, originating from requirements analysis, specification and other activities conducted in the software development. Therefore , software quality estimation is an activity needed at various stages . It may be used for planning the project based quality assurance practices and for benchmarking. In addition, the number of defects per unit is considered one of the most important factors that indicate quality of the software.

Analysis and Design

Objective:

In this application we are giving software quality predictions using machine learning approaches such as XGBoost, Random Forest, Decision Tree, Logistic Regression and Naïve Bayes.

Scope:

- Our Models can be used for detecting the Quality of the software.
- It can be helpful for the customers to use whether they use the Particular software or not.

Existing Method

In previous studies , two methods (Multiple Criteria Linear Programming and Multiple Criteria Quadratic Programming) for estimating the quality of software had been used.Also, C5.0, SVM and Neural network were experimented with for quality estimation. These studies have relatively low accuracy.

DISADVANTAGES:

- Accuracy is low.
- Difficult to handle.

Proposed System

In this study , we aimed to improve estimation accuracy by using relevant features of a large dataset . We used a feature selection method and correlation matrix for reaching higher accuracies . In addition , we have experimented with recent methods shown to be successful for other prediction tasks. Machine learning algorithms such as XGBoost , Random Forest , Decision Tree , Logistic Regression and Naïve Bayes are applied to the data to predict the software quality.

ADVANTAGES:

- Accuracy is high.
- Low complexities.

Requirement Specification

Hardware Configuration:

Client Side:

Ram	4 GB
Hard disk	487.0GB
Processor	1AMD® A9-9420 radeon r5, 5 compute cores 2c+3g × 2

Server side:

Ram	4GB
Hard disk	487.0GB
Processor	1AMD® A9-9420 radeon r5, 5 compute cores 2c+3g × 2

Software specification:

Technology	Machine Learning, Application
Libraries	Pandas, Numpy, Sci-Kit Learn.
Version	Python 3.11
Server side Scripts	Html , css, javascript
Frameworks	Flask
IDE	Pycharm IDE

Modules

1.System

1.1 Store Dataset:

The System stores the dataset given by the user.

1.2 Model Training:

The system takes the data from the user and fed that data to the selected Model.

1.3 Model Predictions:

The system takes the data given by the user and predict the output based on the given data.

2.User

2.1 Load Dataset:

The user can load the dataset he/she want to work on.

2.2 View Dataset:

The User can view the dataset.

2.3 Select model:

User can apply the model to the dataset for accuracy.

2.4 Evaluation:

User can evaluate the model performance.

System Analysis

System Analysis is the process of gathering and collecting interpreting facts,diagnosing the problem and using the information to recommended improvement to the system.Analysis is an activity that encompasses most of the tasks that are collectively called computer system engineering. System engineering and analysis and design. Requirements gathering at the system Level with a small amount of top - level analysis and design.Requirements analysis is the first technical step in software engineering process.

System Analysis Includes

→ The surveying and planning of the system.

→ The study and analysis of existing system.
→ The study and analysis of existing system.
→ The definitions of requirements and priorities for new or improved system. For which a popular synonym is Logical design.

Software Development Life Cycle:

There are various software development approaches defined and designed which are used during the development process of software, these approaches are also referred to as “software development process models”.

Each process model follows a particular life cycle in order to ensure success in process of software development that is called software development life cycle.

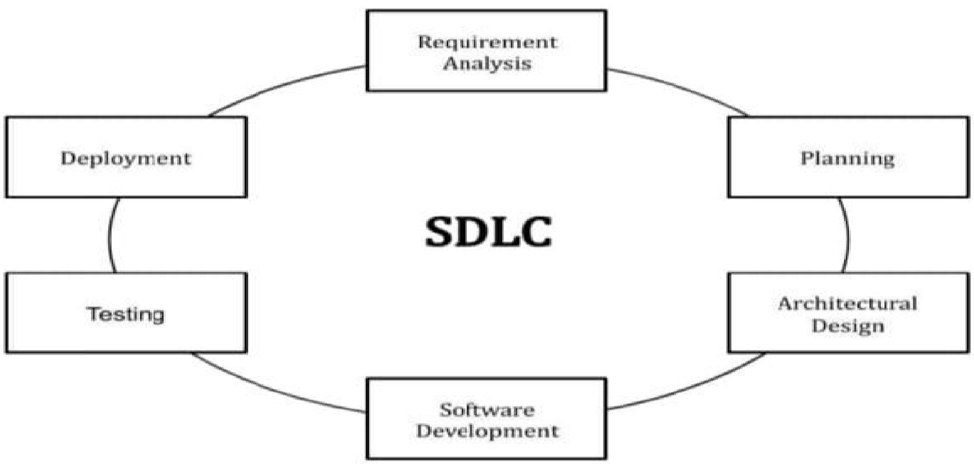


Figure-5.1.1: SDLC life cycle

Planning:

Planning for the quality assurance requirements and identifications of the risks associated with the projects is also done at this stage.

Design:

The software system design is produced from the results of the requirements phase .

Development:

Code is produced from the deliverables of the design phase during implementation and this is the longest phase of the software development life cycle.

Testing:

During testing , the implementation is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase.

Deployment:

Once the software is certified , and no bugs or errors are stated then it is deployed,

System design

Homepage:

Home page of the software quality prediction is displayed. It contain The side navigation bar which shows the pages names model training Visualization , reports ,predictions.

Model training:

Model training page contains the accuracy of the models. We have to select the type of model and submit it to display the accuracy of the model.

Visualization:

Visualization page contains the bar graph of the models and how much accuracy they contain when we select the models in model training, then the visualization page redirects to the web page and displays the bar graph.

Reports:

Reports page contain the classification reports of Logistic existing and KNN proposed**.**

Predictions:

Prediction page contains the ten parameters which explains about the quality of project and they are trained in binary values when the project is good it takes 1 as input if the project quality is bad then it takes 0 as input and predict the quality of the software

Algorithms used

XGBoost Classifier:

Data Preparation: we need to gather data related to software quality, which can include metrics like code complexity, number of bugs, code coverage, etc. Once we have the data , we need to prepare it for analysis by cleaning , transforming, and normalizing it.

1. Building the Decision Tree
2. Pruning the Tree
3. Evaluation
4. Finally, we evaluate the performance of the decision tree classifier on a validation set using metrics like accuracy, precision, recall, and F1 score.

Random Forest Classifier:

1. Data Preparation
2. Building Trees: Next, we create multiple decision trees using different subsets of the data. Each tree is trained on a randomly selected subset of the data, and each split in the tree is made by selecting the feature that maximizes the information gain.
3. Ensemble: Once we have created multiple decision trees, we combine them to create a random forest. The random forest makes predictions by aggregating the predictions of each decision tree. For classification problems, the prediction is based on the mode of the class predictions of the individual trees. For regression problems, the prediction is based on the average of the predictions of the individual trees.
4. Prediction: Once the random forest is trained, it can be used to predict the quality of new software. We simply pass the relevant features of the new software to the random forest, and it returns a predicted quality score.

Decision Tree Classifier:

Data Preparation

Building the Decision Tree

Pruning the Decision Tree

Prediction

Evaluation

Logistic Regression:

Feature Selection :we select a subset of the features that are most predictive of software quality. We can use statistical methods like chi-squared or mutual information to select the most informative features.

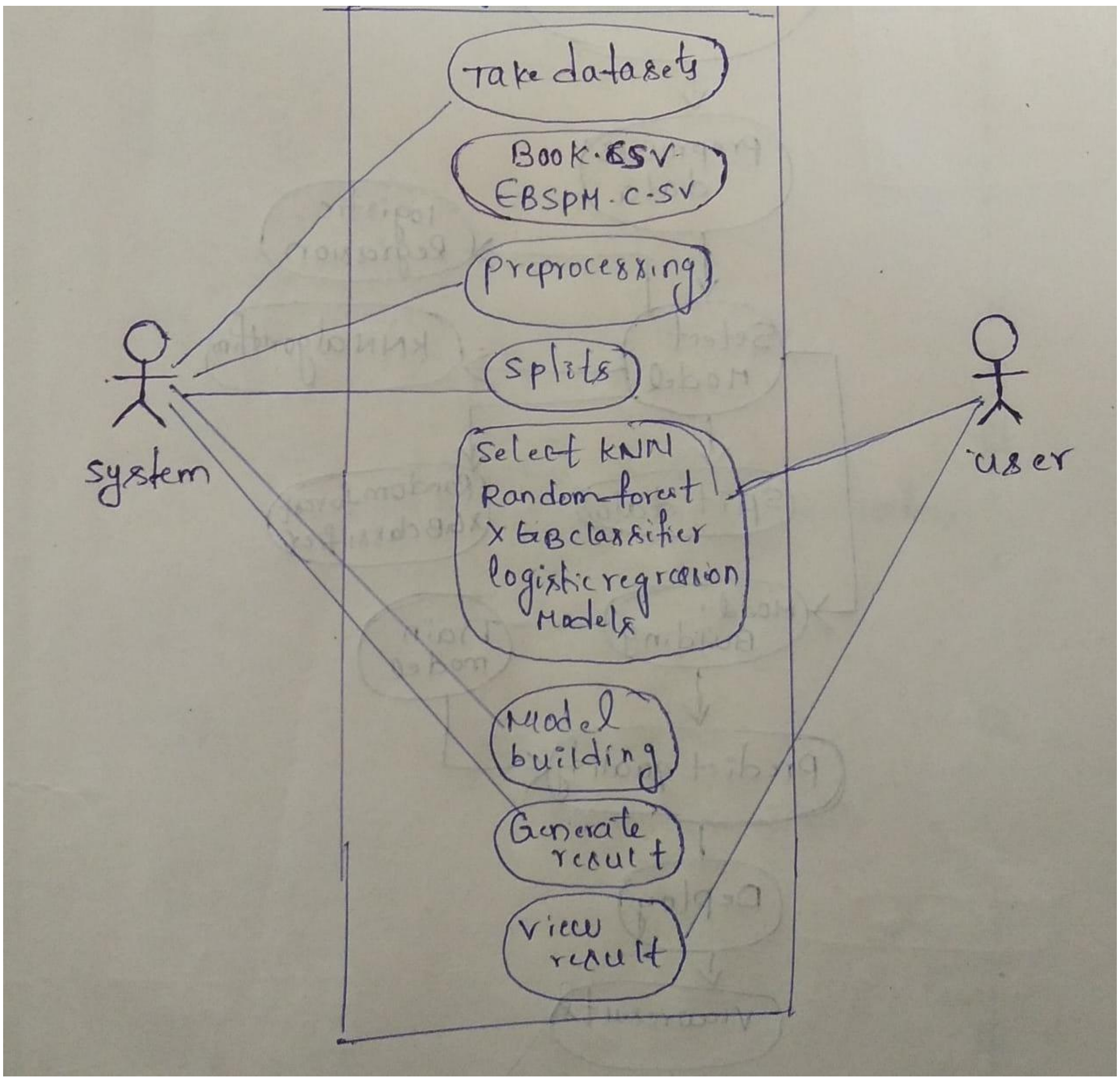
1. Model Training: Once we have selected the features, we train a logistic regression model using the labeled training data. The logistic regression model learns the relationship between the input features and the binary output by estimating the parameters of a logistic function.
2. Prediction: Once the logistic regression model is trained, it can be used to predict the quality of new software . We simply pass the relevant features of the new software to the logistic regression model, and it returns a predicted probability of being high quality or low quality.
3. Evaluation: Finally, we evaluate the performance of the logistic regression model on a validation set using metrics like accuracy, precision, recall, and F1 score.

KNN Algorithm:

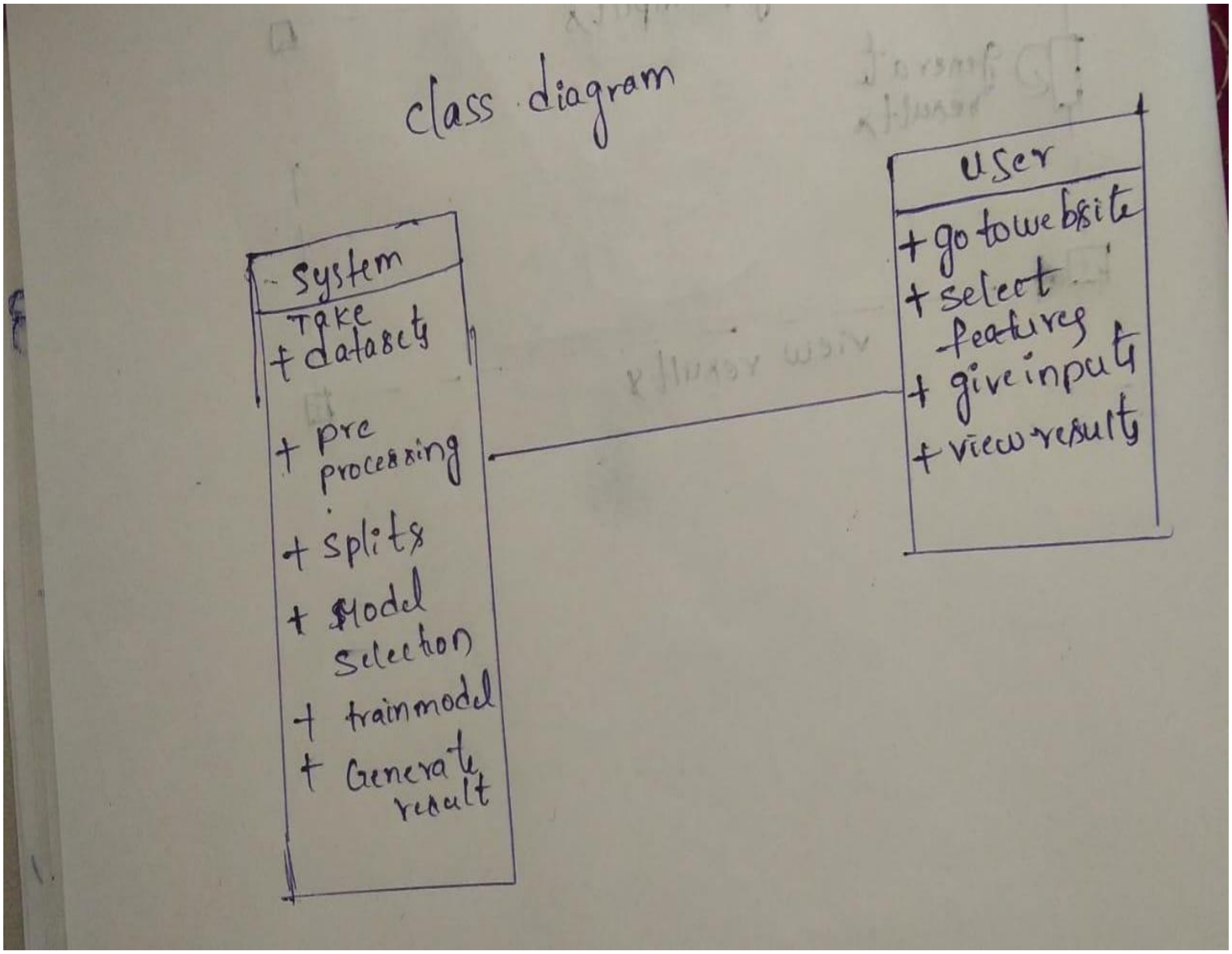
1. Choosing k: We need to choose the value of k, which is the number of nearest neighbors that will be considered when making a prediction. The value of k can be chosen using cross-validation or other methods.
2. Calculating distances: For each new data point, we calculate the distance to all the other data points in the training set. The most common distance metric used is Euclidean distance, but other metrics can also be used.
3. Finding the k-Nearest Neighbors: We select the k-nearest neighbors based on the calculated distances. These are the data points in the training set that are closest to the new data point.
4. Prediction: For classification problems , we make a prediction by taking the majority class of the k-nearest neighbors. For regression problems, we make a prediction by taking the average value of the k-nearest neighbors.

5. Evaluation: Finally, we evaluate the performance of the k-NN algorithm on a validation set using metrics like accuracy, precision, recall, and F1 score.

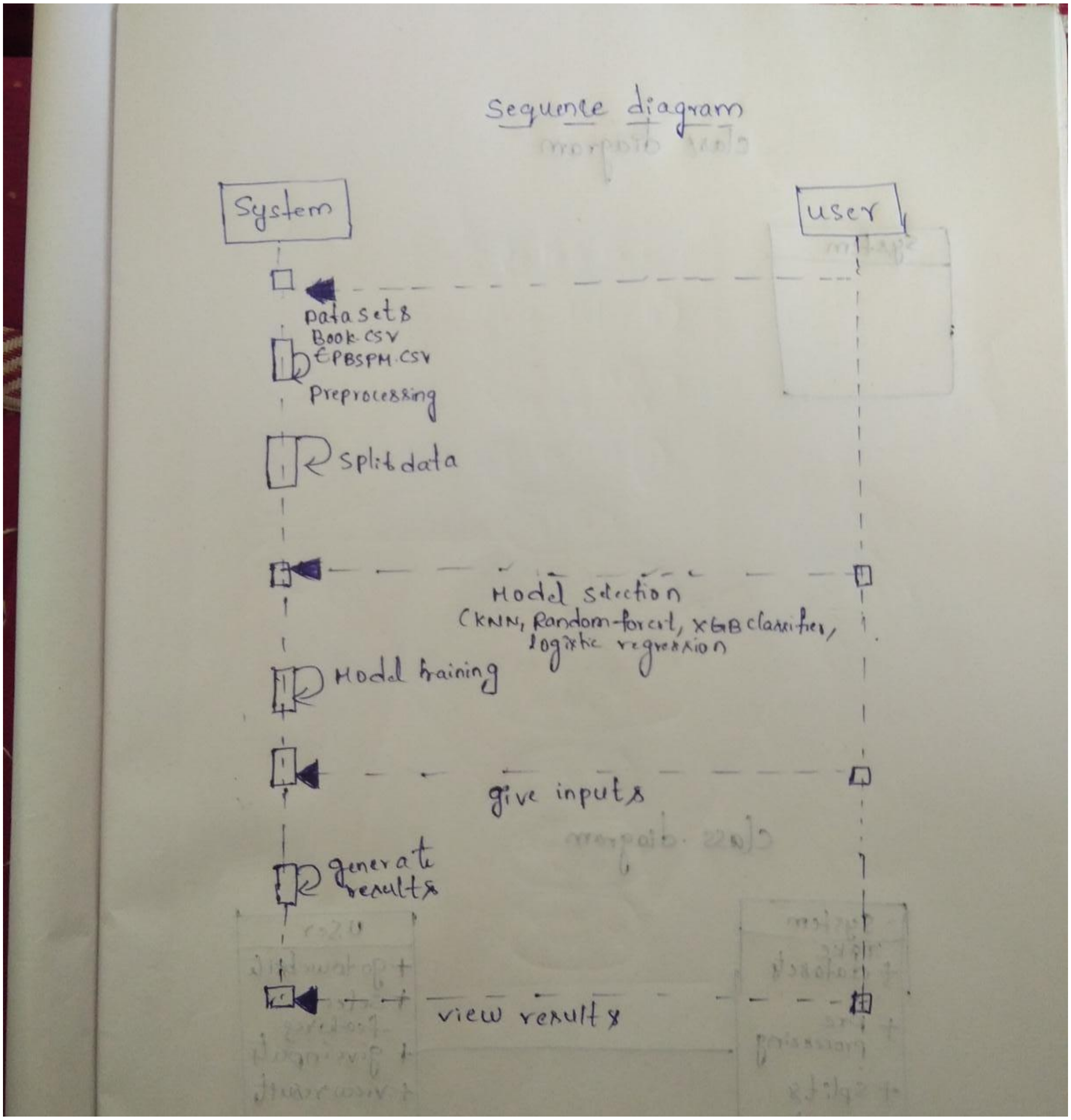
UML DIAGRAMS



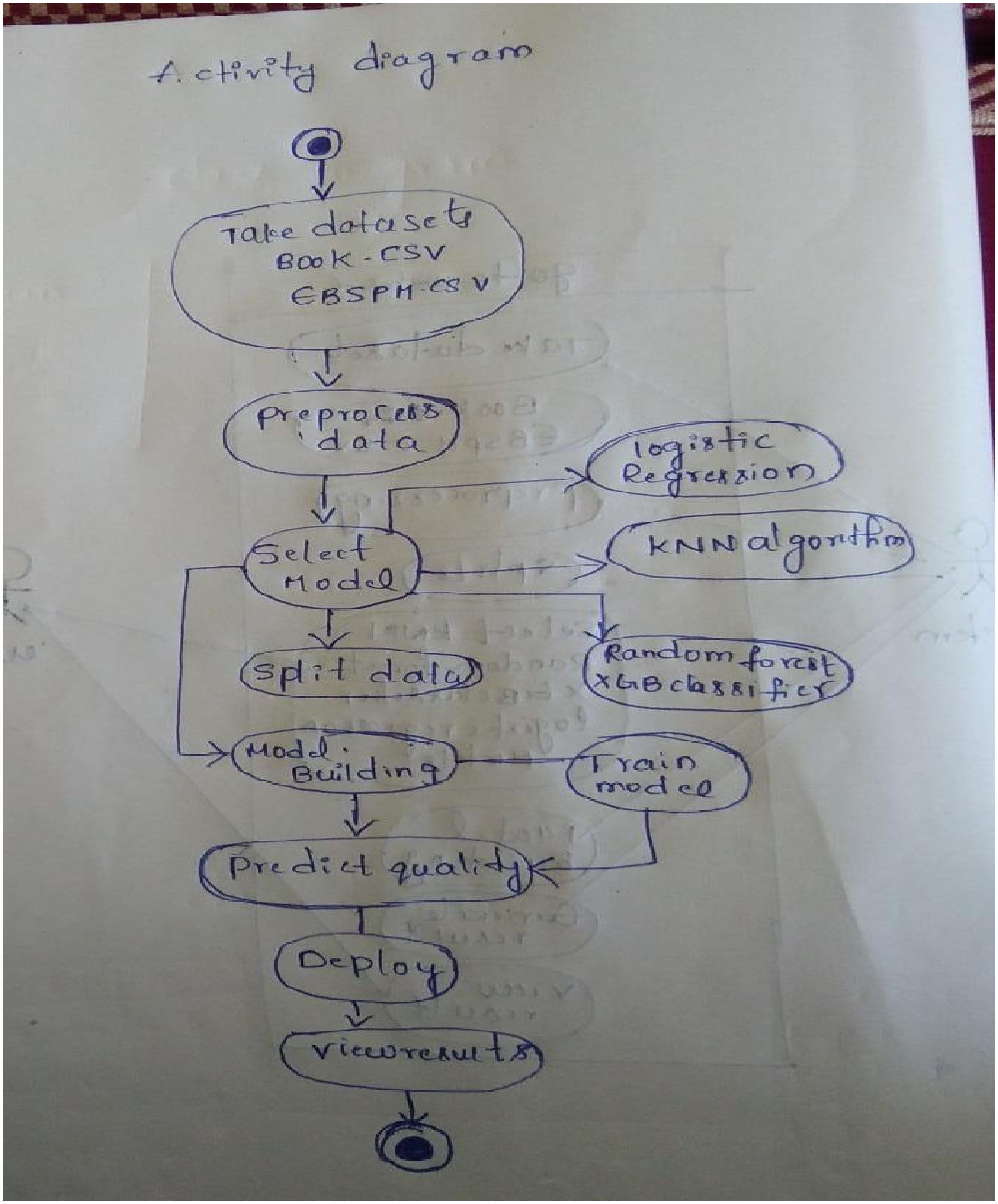
Class diagram



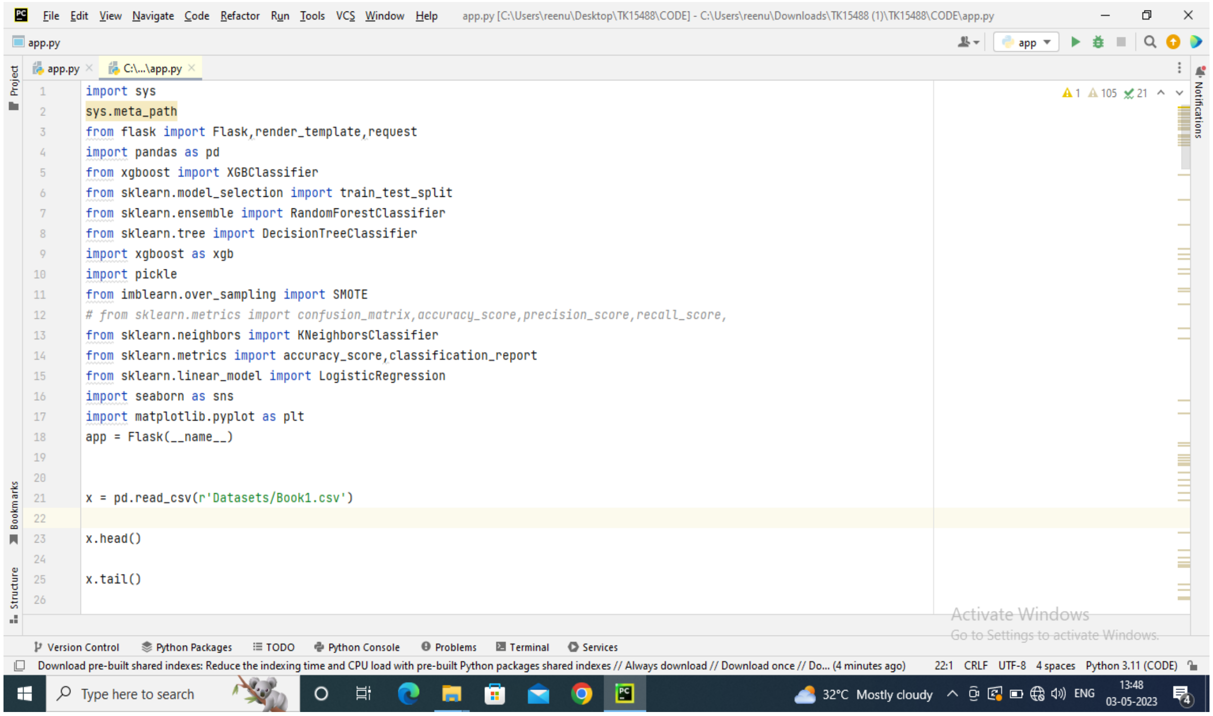
SEQUENCE DIAGRAM



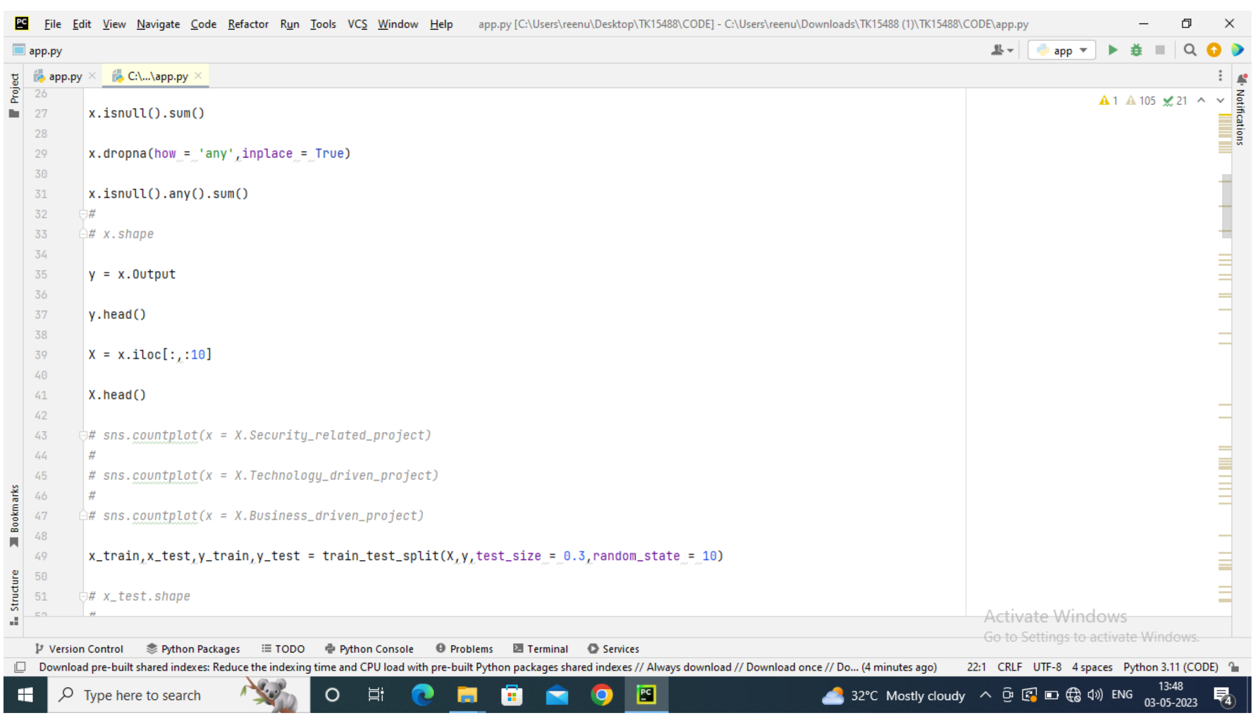
ACTIVITY DIAGRAM



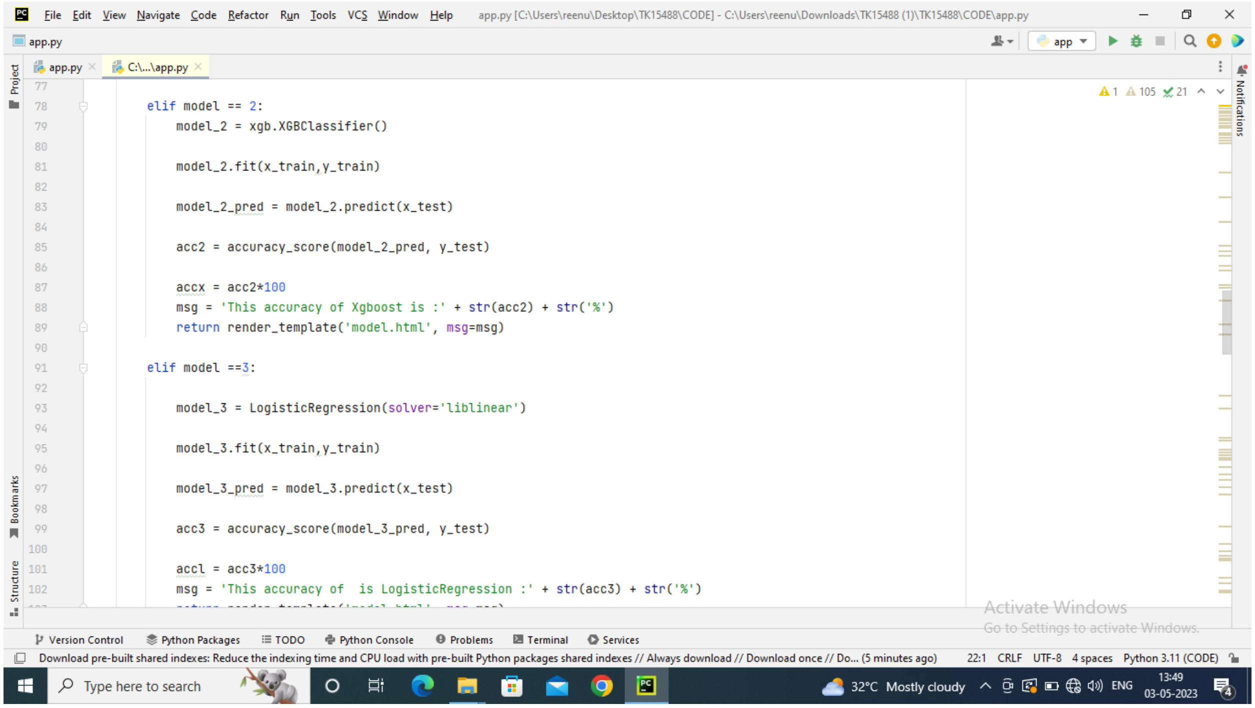
SOURCE CODE



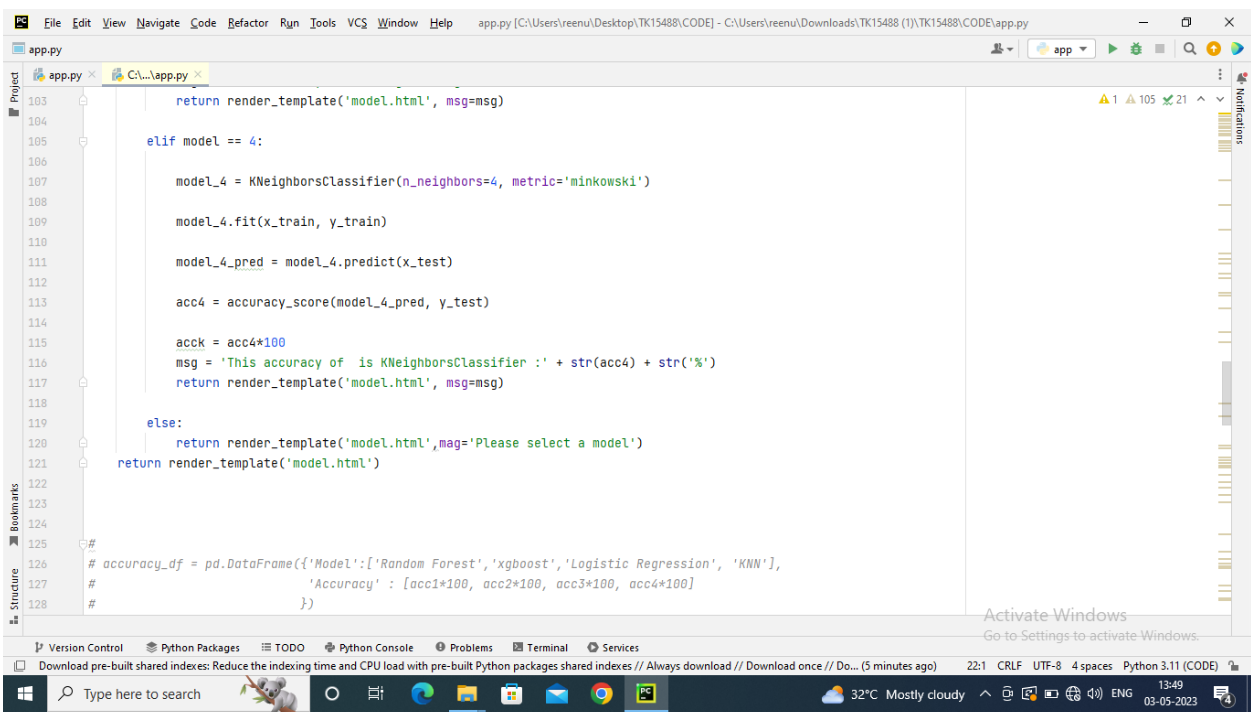
```python
import sys
sys.meta_path
from flask import Flask,render_template,request
import pandas as pd
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
import xgboost as xgb
import pickle
from imblearn.over_sampling import SMOTE
# from sklearn.metrics import confusion_matrix,accuracy_score,precision_score,recall_score,
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score,classification_report
from sklearn.linear_model import LogisticRegression
import seaborn as sns
import matplotlib.pyplot as plt
app = Flask(__name__)


x = pd.read_csv(r'Datasets/Book1.csv')

x.head()

x.tail()
```



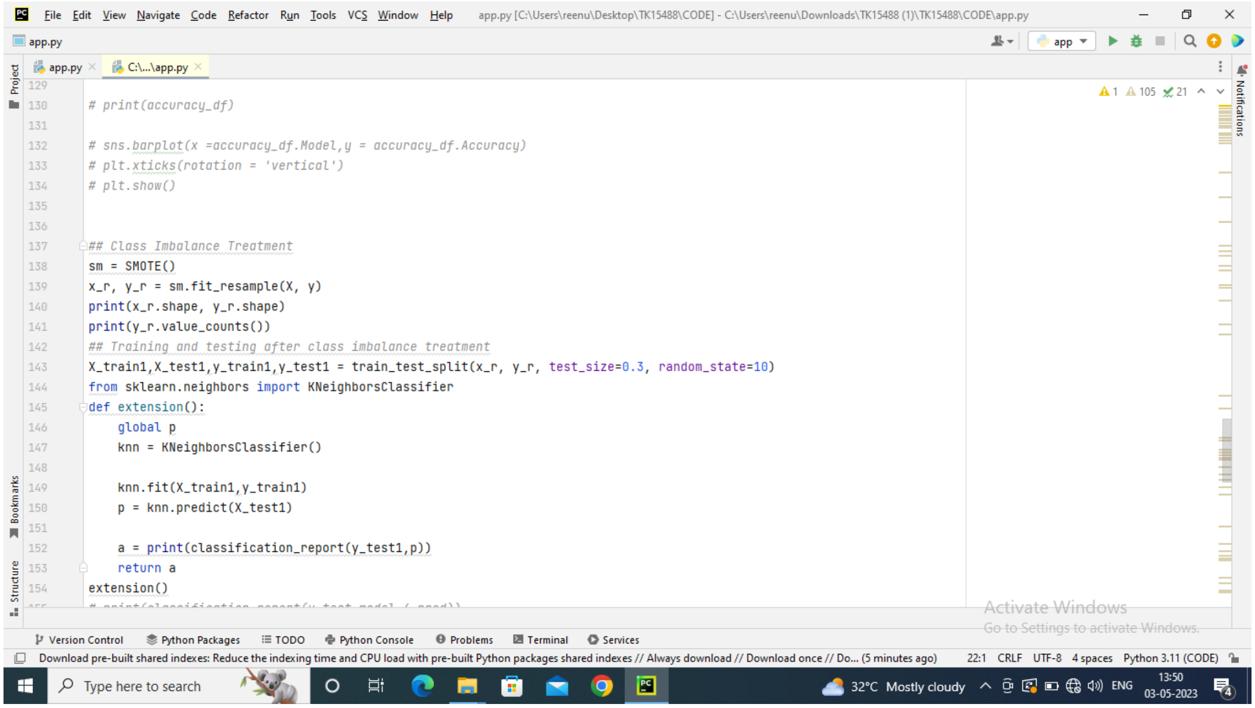
```python
x.isnull().sum()

x.dropna(how = 'any',inplace = True)

x.isnull().any().sum()
#
# x.shape

y = x.Output

y.head()

X = x.iloc[:,:10]

X.head()

# sns.countplot(x = X.Security_related_project)
#
# sns.countplot(x = X.Technology_driven_project)
#
# sns.countplot(x = X.Business_driven_project)

x_train,x_test,y_train,y_test = train_test_split(X,y,test_size = 0.3,random_state = 10)

# x_test.shape
```



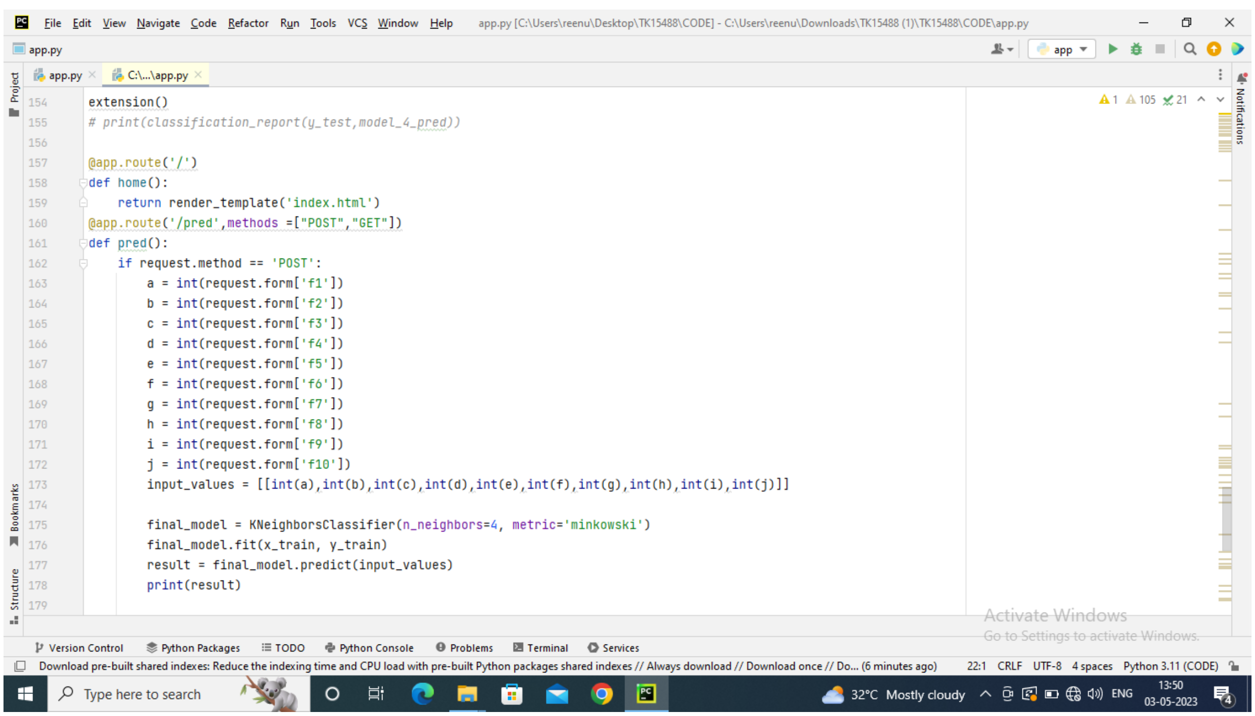
```
    elif model == 2:
        model_2 = xgb.XGBClassifier()

        model_2.fit(x_train,y_train)

        model_2_pred = model_2.predict(x_test)

        acc2 = accuracy_score(model_2_pred, y_test)

        accx = acc2*100
        msg = 'This accuracy of Xgboost is :' + str(acc2) + str('%')
        return render_template('model.html', msg=msg)

    elif model ==3:

        model_3 = LogisticRegression(solver='liblinear')

        model_3.fit(x_train,y_train)

        model_3_pred = model_3.predict(x_test)

        acc3 = accuracy_score(model_3_pred, y_test)

        accl = acc3*100
        msg = 'This accuracy of  is LogisticRegression :' + str(acc3) + str('%')
```

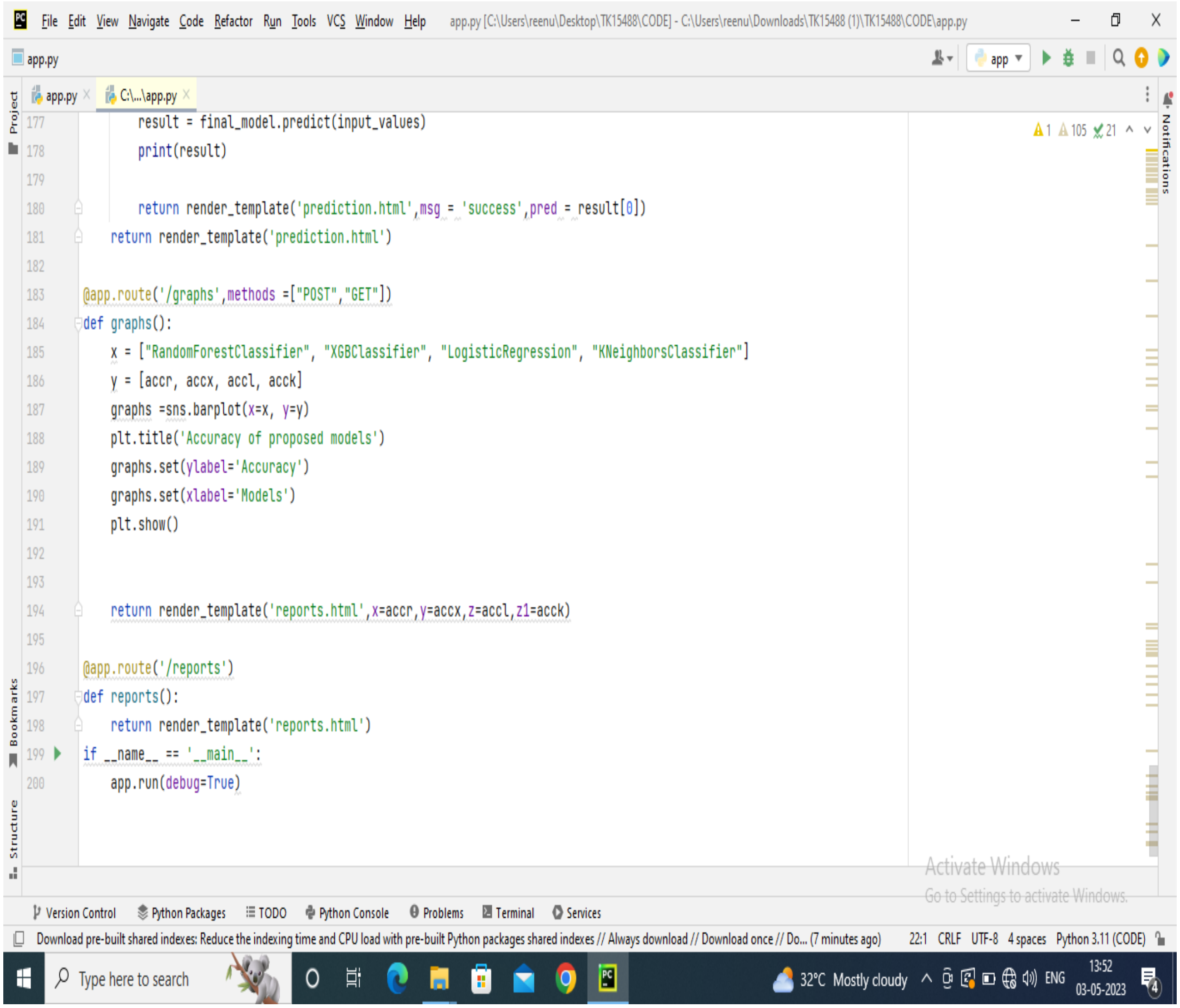
```
        return render_template('model.html', msg=msg)

    elif model == 4:

        model_4 = KNeighborsClassifier(n_neighbors=4, metric='minkowski')

        model_4.fit(x_train, y_train)

        model_4_pred = model_4.predict(x_test)

        acc4 = accuracy_score(model_4_pred, y_test)

        acck = acc4*100
        msg = 'This accuracy of  is KNeighborsClassifier :' + str(acc4) + str('%')
        return render_template('model.html', msg=msg)

    else:
        return render_template('model.html',mag='Please select a model')
    return render_template('model.html')



#
# accuracy_df = pd.DataFrame({'Model':['Random Forest','xgboost','Logistic Regression', 'KNN'],
#                       'Accuracy' : [acc1*100, acc2*100, acc3*100, acc4*100]
#                       })
```



```python
# print(accuracy_df)

# sns.barplot(x =accuracy_df.Model,y = accuracy_df.Accuracy)
# plt.xticks(rotation = 'vertical')
# plt.show()


## Class Imbalance Treatment
sm = SMOTE()
x_r, y_r = sm.fit_resample(X, y)
print(x_r.shape, y_r.shape)
print(y_r.value_counts())
## Training and testing after class imbalance treatment
X_train1,X_test1,y_train1,y_test1 = train_test_split(x_r, y_r, test_size=0.3, random_state=10)
from sklearn.neighbors import KNeighborsClassifier
def extension():
    global p
    knn = KNeighborsClassifier()

    knn.fit(X_train1,y_train1)
    p = knn.predict(X_test1)

    a = print(classification_report(y_test1,p))
    return a
extension()
```

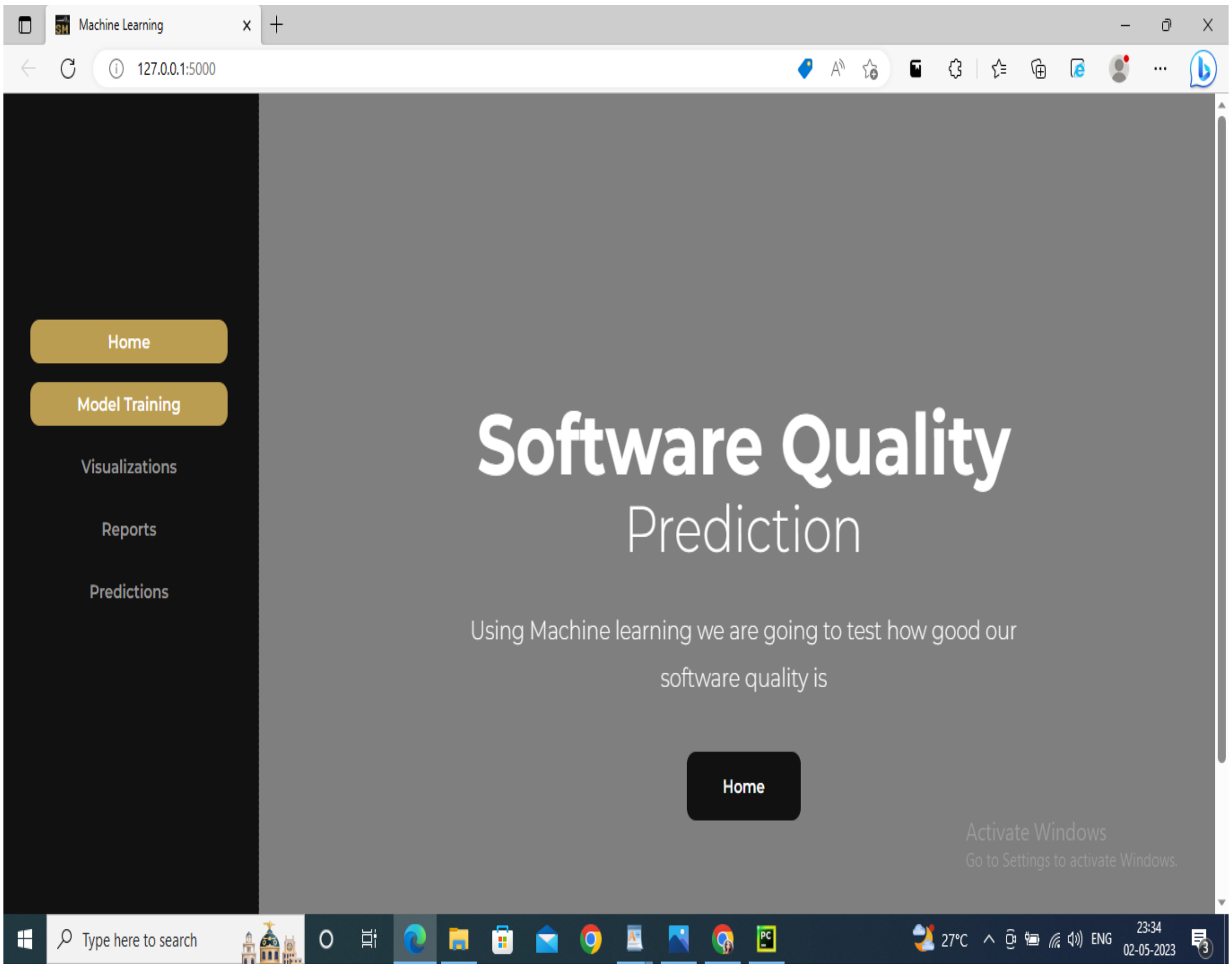


```python
extension()
# print(classification_report(y_test,model_4_pred))

@app.route('/')
def home():
    return render_template('index.html')
@app.route('/pred',methods =["POST","GET"])
def pred():
    if request.method == 'POST':
        a = int(request.form['f1'])
        b = int(request.form['f2'])
        c = int(request.form['f3'])
        d = int(request.form['f4'])
        e = int(request.form['f5'])
        f = int(request.form['f6'])
        g = int(request.form['f7'])
        h = int(request.form['f8'])
        i = int(request.form['f9'])
        j = int(request.form['f10'])
        input_values = [[int(a),int(b),int(c),int(d),int(e),int(f),int(g),int(h),int(i),int(j)]]

        final_model = KNeighborsClassifier(n_neighbors=4, metric='minkowski')
        final_model.fit(x_train, y_train)
        result = final_model.predict(input_values)
        print(result)
```

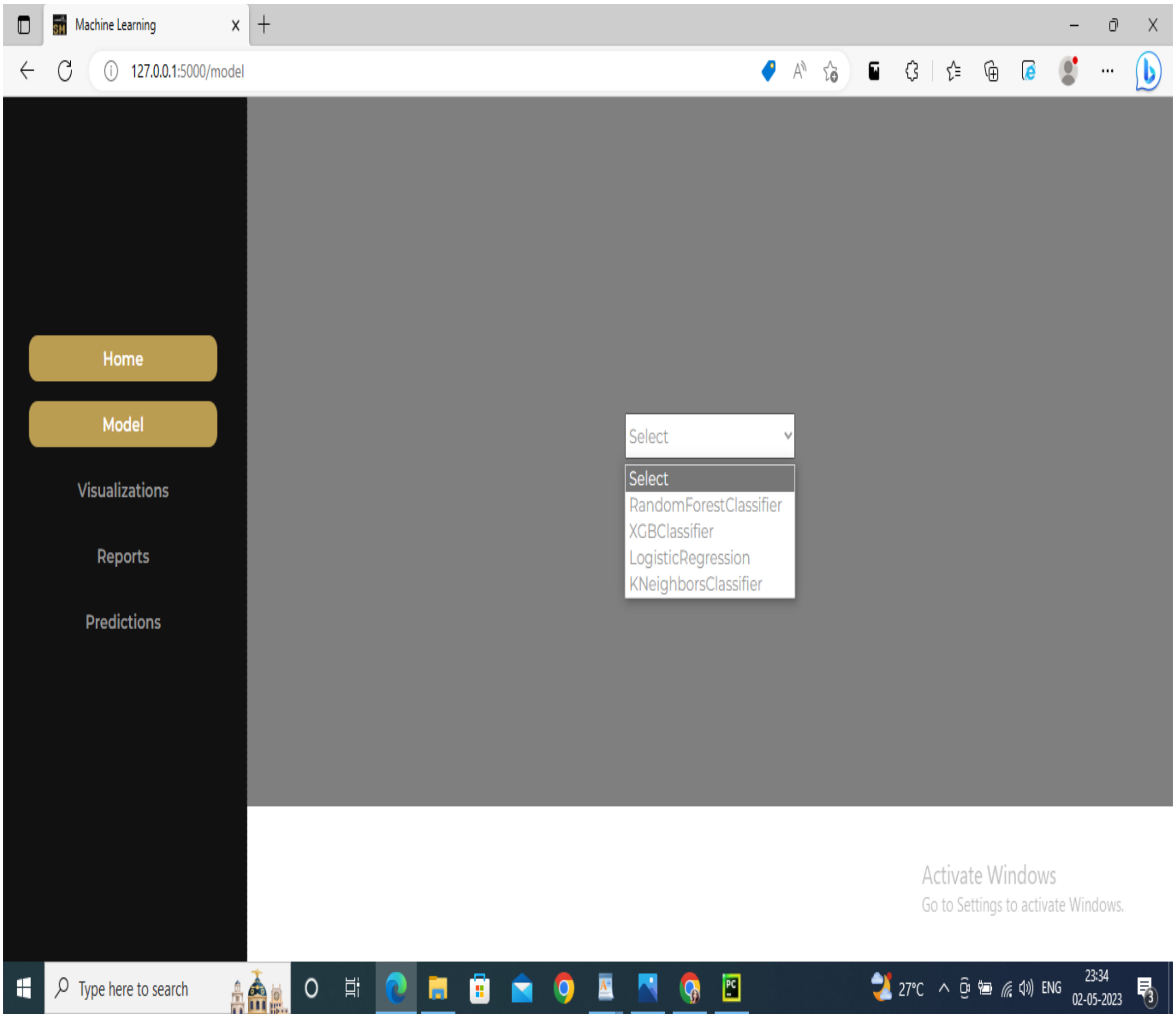

File Edit View Navigate Code Refactor Run Tools VCS Window Help app.py [C:\Users\reenu\Desktop\TK15488\CODE] - C:\Users\reenu\Downloads\TK15488 (1)\TK15488\CODE\app.py
app.py
app.py
C:\...\app.py
result = final_model.predict(input_values)
print(result)
return render_template('prediction.html',msg = 'success',pred = result[0])
return render_template('prediction.html')
@app.route('/graphs',methods =["POST","GET"])
def graphs():
x = ["RandomForestClassifier", "XGBClassifier", "LogisticRegression", "KNeighborsClassifier"]
y = [accr, accx, accl, acck]
graphs =sns.barplot(x=x, y=y)
plt.title('Accuracy of proposed models')
graphs.set(ylabel='Accuracy')
graphs.set(xlabel='Models')
plt.show()
return render_template('reports.html',x=accr,y=accx,z=accl,z1=acck)
@app.route('/reports')
def reports():
return render_template('reports.html')
if __name__ == '__main__':
app.run(debug=True)
Activate Windows
Go to Settings to activate Windows.
Version Control
Python Packages
TODO
Python Console
Problems
Terminal
Services
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Do... (7 minutes ago)
22:1 CRLF UTF-8 4 spaces Python 3.11 (CODE)
Type here to search
32°C Mostly cloudy
ENG
13:52
03-05-2023

OUTPUT

Home page:



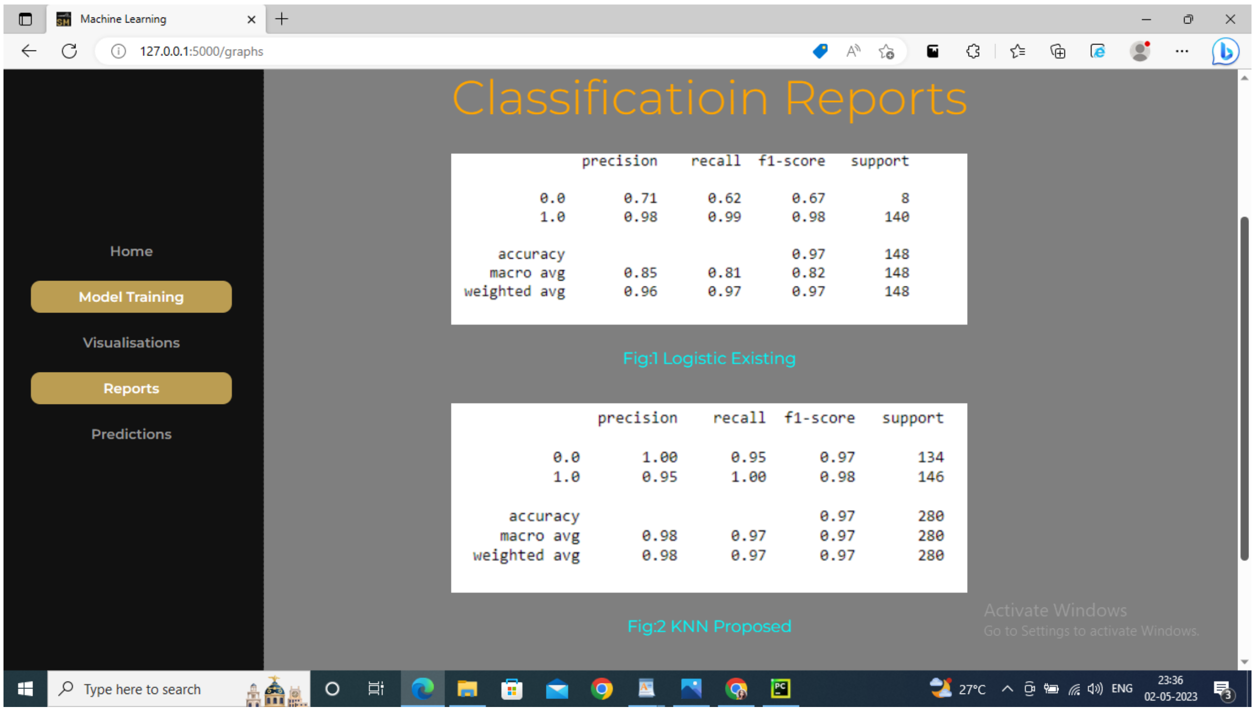
Model page:



Machine Learning
127.0.0.1:5000/model
Home
Model
Visualizations
Reports
Predictions
Select
Select
RandomForestClassifier
XGBClassifier
LogisticRegression
KNeighborsClassifier
Activate Windows
Go to Settings to activate Windows.
Type here to search
27°C
ENG
23:34
02-05-2023

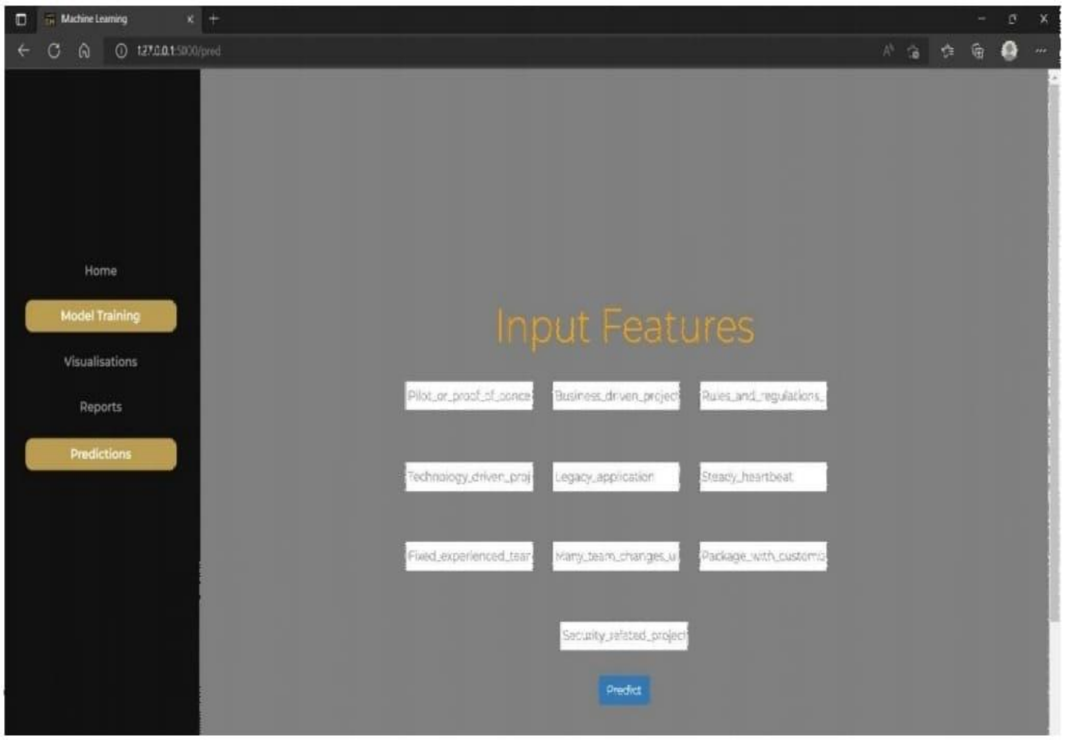
Visualization page:

Report page:


Machine Learning
127.0.0.1:5000/graphs
Classificatioin Reports
Home
Model Training
Visualisations
Reports
Predictions
precision recall f1-score support
0.0 0.71 0.62 0.67 8
1.0 0.98 0.99 0.98 140
accuracy 0.97 148
macro avg 0.85 0.81 0.82 148
weighted avg 0.96 0.97 0.97 148
Fig:1 Logistic Existing
precision recall f1-score support
0.0 1.00 0.95 0.97 134
1.0 0.95 1.00 0.98 146
accuracy 0.97 280
macro avg 0.98 0.97 0.97 280
weighted avg 0.98 0.97 0.97 280
Fig:2 KNN Proposed
Activate Windows
Go to Settings to activate Windows.
Type here to search
27°C
ENG
23:36
02-05-2023

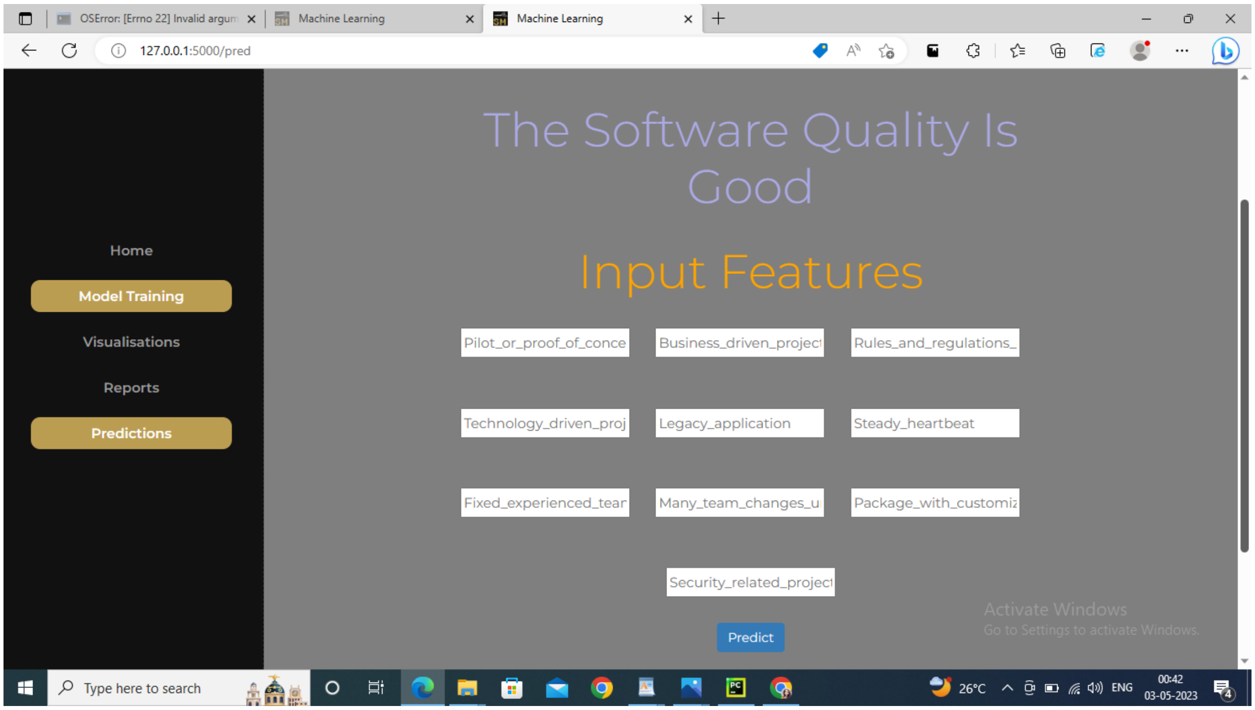
Input features page:

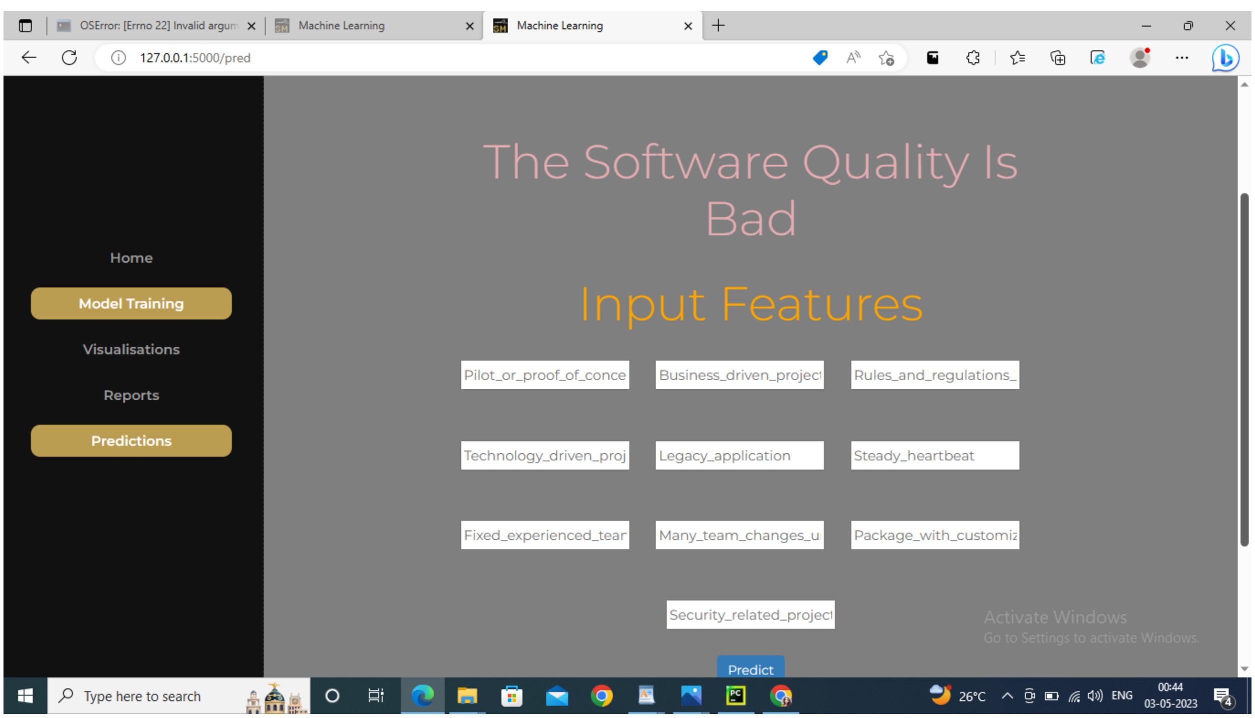
INPUT FEATURES PAGE



8.1.6 Input Features Page

Final output page:





Conclusion

In this application,we used supervised Machine learning models to predict the quality of the software.Total five ML algorithms to predict software quality are random forest classifier, decision tree classifier, Xgboost classifier, KNN all algorithms performs well with good accuracies.