נספח:

model_builder.py

class MainScreen(tkinter.Tk)

```
|  MainScreen(*args, **kwargs)
|
|  the class for the model builder and runner frontend and partly backend.
|  shows a gui for the user to choose the csv file how to clean it and the model to run.
|  the software creates clean csv, model file, runs the model
|  saves the results as png files of confusion matrix and finally on Excel file.
|
|   Fields defined here:
|
|   fill_dict  -  dictionary for the filling missing values functions
|   browse_button  -  button component for the browse file option
|   column_selected  -  tk variable for the classify column name selected
|   classify_combox_selection  -  combobox component for selecting the classify column
|   missing_value_selected  -  tk variable of the filling values method
|   missing_values_combox_selection  -  combobox component for choosing the filling the missing values method
|   normalization_selected  -  tk variable of if normalize is needed or not
|   normalization_combox_selection  -  combobox component for choosing if normalize
|   discretization_selected  -  tk variable of the discretize method chosen
|   discretization_combox_selection  -  combobox component for choosing the descretize methode
|   bins_selected  -  tk variable of the amount of bins selected
|   bins_entry_selection  -  entry component for inserting how many bins needed
|   model_selected  -  tk variable of the model chosen
|   model_combox_selection  -  combobox compomemt for choosing the model
|   implement_selected  -  tk variable of the implement of the model chosen
|   model_combox_implement  -  combobox component for choosing the implement
|   build_button  -  button component for building the model and cleaning the file
|   run_button  -  button component for running the model and saving the results
|   file  -  the csv file chosen
|   filePath  -  the chosen csv file's path
|   class_values  -  the classify column unique values
|   data_train  -  the data needed to train the model
|   data_test  -  the data needed to test the model
|   class_train  -  the classify data of the train data
|   class_test  -  the classify data of the test data
|   train_prediction  -  the prediction of the train data
|   test_prediction  -  the prediction of the test data
|   majority_law_array  -  array of the majority law
|
|
|  Methods defined here:
|
|  __init__(self, *args, **kwargs)
|     init and fill the window with the components
|
|  browseFiles(self)
|     open a file dialog to choose the csv file.
|     open the chosen file and if not empty, get the columns names for later use.
|
|  buildModel(self)
```

| encode the data, split it to train and test.

| build the model by the choices of the user and save it as a new file using pickle.

|

| **<u>cleanAndBuildModel(self)</u>**

| clean the csv file and build the model.

| save the model and the clean csv file as new files.

|

| **<u>cleanFile(self)</u>**

| clean the csv file by the choices of the user and save the clean csv it as a new file.

|

| **<u>fillResults(self)</u>**

| fill the results Excel file with the results of the model and the cleaning settings.

|

| **<u>isDiscretAndBinsOk(self)</u>**

| check if discretization is chosen and if bins is inserted (if not chosen without discretization).

| :return: if the discretization and bins selected is valid or not.

|

| **<u>onModelSelect(self, x=None)</u>**

| when the model is selected at the combobox, if needed enable the implementation combox.

|

| **<u>runModel(self)</u>**

| open and run the model and save the results as png of the confusion Matrix and creates the majority law array

model_components.py

all necessary functions and class to clean csv file, create and train and predict model and create confusion matrix. The frontend

contains:

class selfNaiveBayes(builtins.object)

```
|     selfNaiveBayes(data_train, class_train)
|
|     class that builds self made naive bayes model and predict by new data
|
|     Fields defined here:
|
|     data_train  -  the data to train the model
|     class_train  -  the classify data to train the model
|     bayesCalcs  -  all the calculations of the model as a dictionary with the values as key
|     pClass  -  the probability of every value in the classify
|
|
|     Methods defined here:
|
|     __init__(self, data_train, class_train)
|         init the naive bayes with the train data
|         :param data_train: the train data
|         :param class_train: the train classify column
|
|     calcBayes(self, *args)
|        calculate the bayes probability
|        :param args: the data (row of data from data_test)
|         :return: the predicted classify
```

KNN(data_train, class_train)

builds sklearn's kNN model

:param data_train: the train data

:param class_train: the train classify column

:return: the trained model

SplitTrainTest(data_columns, classify_column)

split the data into train and test.

:param data_columns: the data (encoded)

:param classify_column: the classify column (encoded)

:return: the train and test data as data columns and classify column each

__clean(file, classify)

clean the data and transfer the dtype to int or float if needed

:param file: the data

:param classify: the classify column name

confusionMatrix(train_pred_array, test_pred_array, class_train, class_test, model_name, axis_values)

saves the confusion matrixs of the train and test as new png file

:param class_train: the train classify column

:param test_pred_array: the prediction of the test data

:param train_pred_array: the prediction of the train data

:param axis_values: the values needed for the axis (unique values of the classify column)

:param model_name: the name of the model used to predict

:param class_test: the test classify column

__discretize(file, classify, bins, desc_type)

discretize the data by the methode given in desc_type with the number of bins given in bins

:param file: the data

:param classify: the classify column name

:param bins: the number of bins

:param desc_type: the type of discretization needed

__encodeAndPopClass(file, classify)

encode the data and pop the classify column.

:param file: the data

:param classify: the classify column name

:return: the encoded data and the encoded classify column

__fillByAll(file, classify)

fill missing values by the value of all rows

:param file: the data

:param classify: the classify column name

__fillByClass(file, classify)

fill missing values by the values of the rows with the same classify value

:param file: the data

:param classify: the classify column name

__kMeans(data_train, class_train)

builds sklearn's k-means model

:param data_train: the train data

:param class_train: the train classify column

:return: the trained model

__normalize(file, classify)

normalize the data

:param file: the data

:param classify: the classify column name

__predict(model, data_train, data_test, implement)

predict the classify with the model given in model param and the data given in data_test param

and the implement given in implement param

:param data_train: the train data

:param implement: kind of implement

:param model: the trained model

:param data_test: the test data

:return: the prediction of the model as numpy array

selfDecisionTree(data_train, class_train, data_test)

self decision tree

__skDecisionTree(data_train, class_train)

builds sklearn's decision tree model

:param data_train: the train data

:param class_train: the train classify column

:return: the trained model

__skNaiveBayes(data_train, class_train)

builds sklearn's naive bayes model

:param data_train: the train data

:param class_train: the train classify column

:return: the trained model