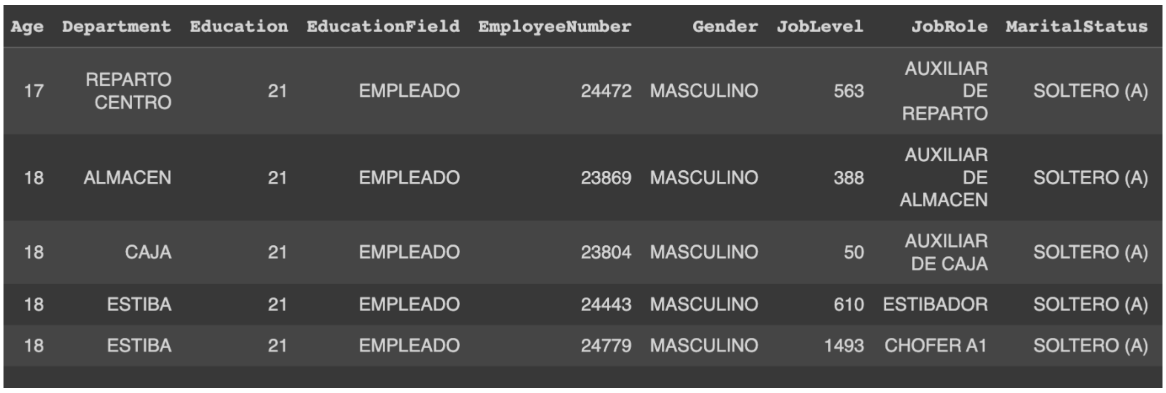
Employee Resignation Project Walk Through

1. Machine Learning

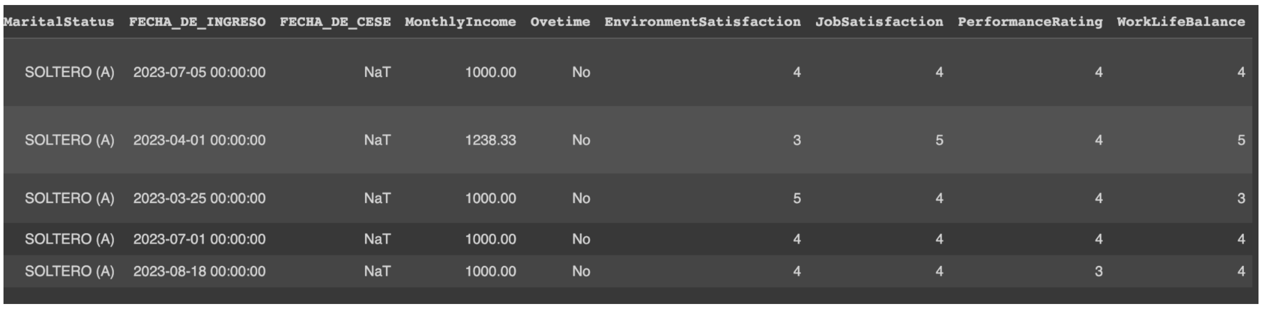
Machine learning part mainly consists of three parts. The first is to preprocess data to make it ready for the ML model and the second is to apply a random forest model, lastly evaluate model results

a. Data Preprocessing

The raw data contains string values, categorical columns as well as the datetime, values are not suitable for machine learning model for that we need to convert them into one integer values to make it readable for random forest model.

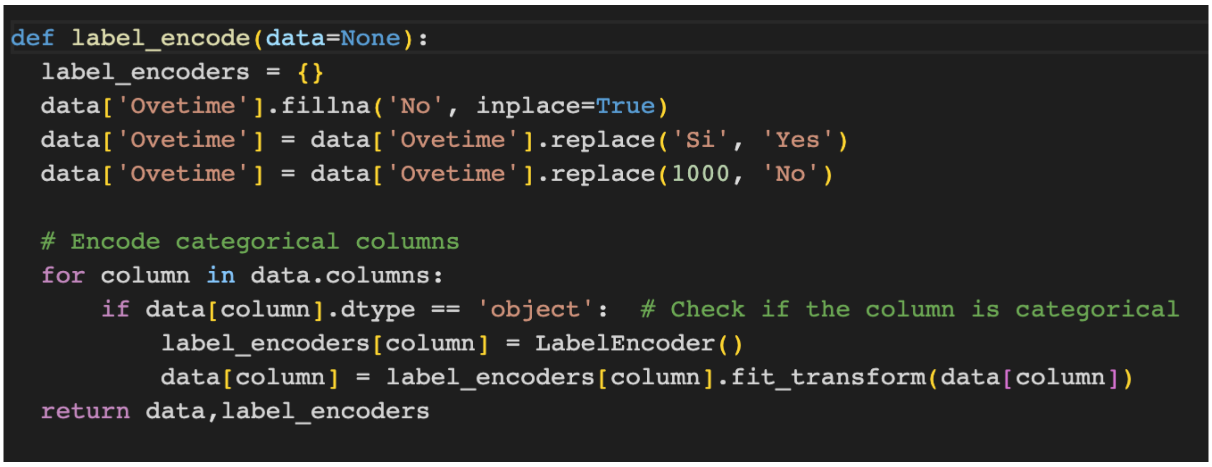


Age	Department	Education	EducationField	EmployeeNumber	Gender	JobLevel	JobRole	MaritalStatus
17	REPARTO CENTRO	21	EMPLEADO	24472	MASCULINO	563	AUXILIAR DE REPARTO	SOLTERO (A)
18	ALMACEN	21	EMPLEADO	23869	MASCULINO	388	AUXILIAR DE ALMACEN	SOLTERO (A)
18	CAJA	21	EMPLEADO	23804	MASCULINO	50	AUXILIAR DE CAJA	SOLTERO (A)
18	ESTIBA	21	EMPLEADO	24443	MASCULINO	610	ESTIBADOR	SOLTERO (A)
18	ESTIBA	21	EMPLEADO	24779	MASCULINO	1493	CHOFER A1	SOLTERO (A)



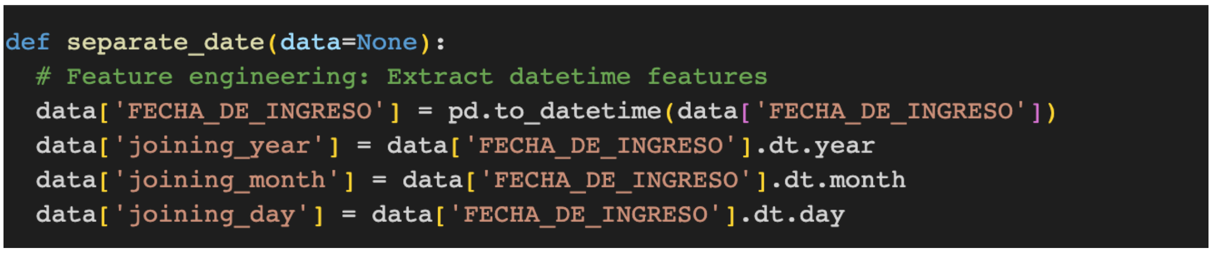
MaritalStatus	FECHA_DE_INGRESO	FECHA_DE_CESE	MonthlyIncome	Ovetime	EnvironmentSatisfaction	JobSatisfaction	PerformanceRating	WorkLifeBalance
SOLTERO (A)	2023-07-05 00:00:00	NaT	1000.00	No	4	4	4	4
SOLTERO (A)	2023-04-01 00:00:00	NaT	1238.33	No	3	5	4	5
SOLTERO (A)	2023-03-25 00:00:00	NaT	1000.00	No	5	4	4	3
SOLTERO (A)	2023-07-01 00:00:00	NaT	1000.00	No	4	4	4	4
SOLTERO (A)	2023-08-18 00:00:00	NaT	1000.00	No	4	4	3	4

After reading the data into Pandas data frame we need to apply label encodings from sklearn to convert string into integer columns



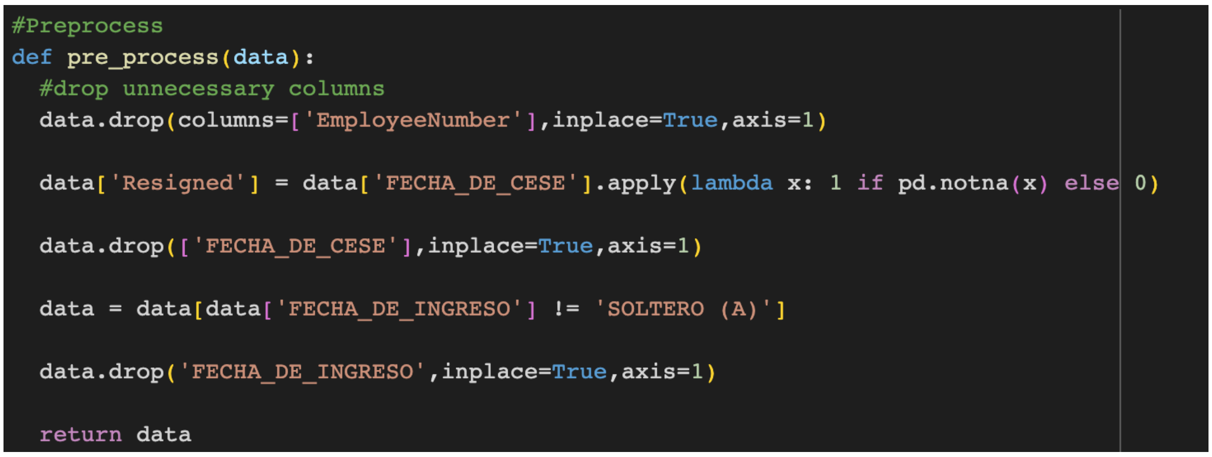
```python
def label_encode(data=None):
  label_encoders = {}
  data['Ovetime'].fillna('No', inplace=True)
  data['Ovetime'] = data['Ovetime'].replace('Si', 'Yes')
  data['Ovetime'] = data['Ovetime'].replace(1000, 'No')

  # Encode categorical columns
  for column in data.columns:
      if data[column].dtype == 'object':  # Check if the column is categorical
          label_encoders[column] = LabelEncoder()
          data[column] = label_encoders[column].fit_transform(data[column])
  return data,label_encoders
```

Another important column is employee joining date which was not in the format suitable for a machine learning model so we separate the date into three different columns that is day, month and year.



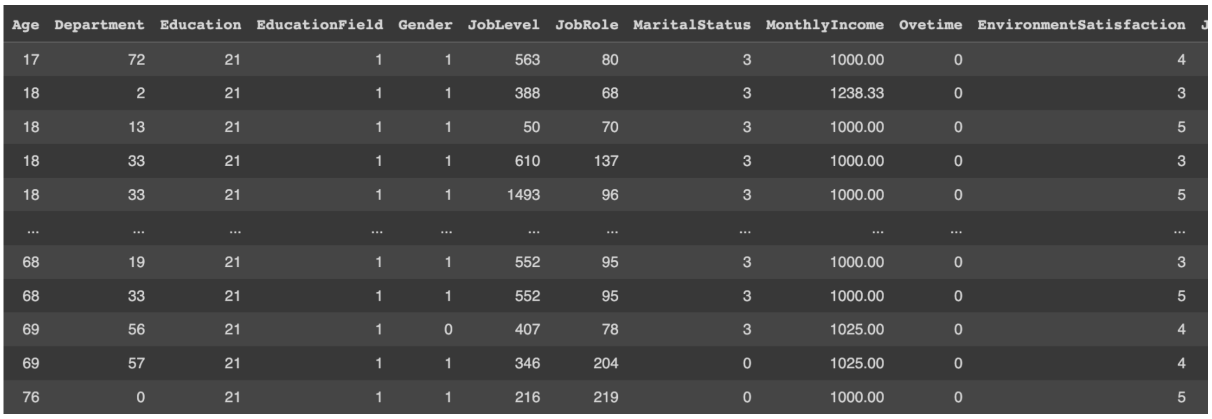
```python
def separate_date(data=None):
  # Feature engineering: Extract datetime features
  data['FECHA_DE_INGRESO'] = pd.to_datetime(data['FECHA_DE_INGRESO'])
  data['joining_year'] = data['FECHA_DE_INGRESO'].dt.year
  data['joining_month'] = data['FECHA_DE_INGRESO'].dt.month
  data['joining_day'] = data['FECHA_DE_INGRESO'].dt.day
```

After label encodings there were some null rows that needs to drop to prevent model under fitting. Also unnecessary columns are also dropped by using drop function from Pandas library for example employee number was just a number that doesn't make any sense for employee resignation and which leads to meaning less results also the target column was missing from the original data so we need to map that column based on the termination date present on the original data

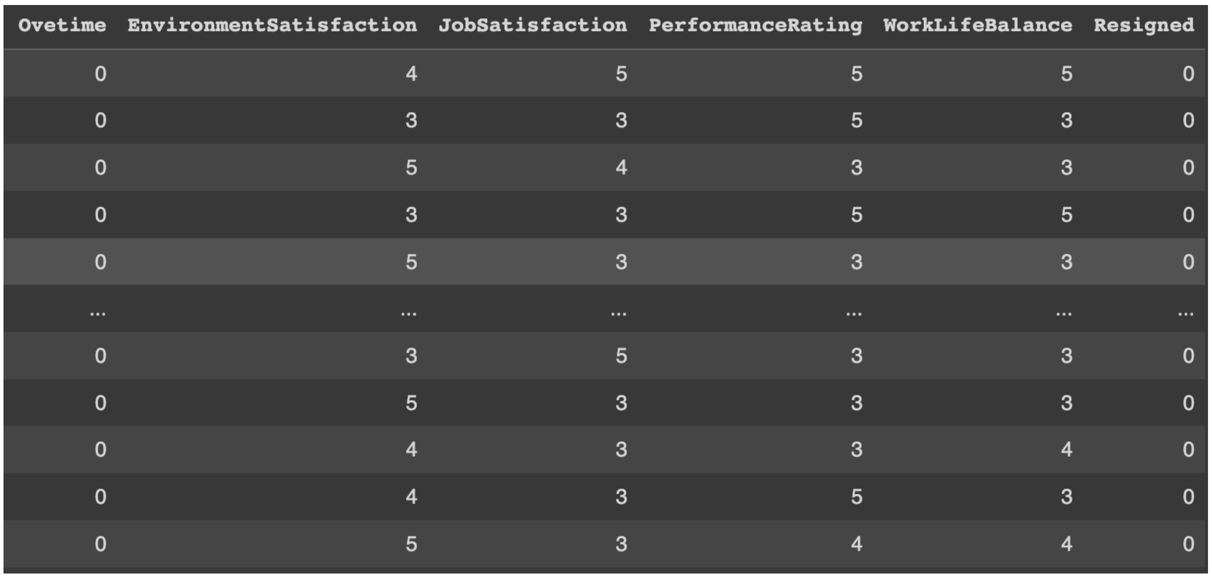


```python
#Preprocess
def pre_process(data):
  #drop unnecessary columns
  data.drop(columns=['EmployeeNumber'],inplace=True,axis=1)

  data['Resigned'] = data['FECHA_DE_CESE'].apply(lambda x: 1 if pd.notna(x) else 0)

  data.drop(['FECHA_DE_CESE'],inplace=True,axis=1)

  data = data[data['FECHA_DE_INGRESO'] != 'SOLTERO (A)']

  data.drop('FECHA_DE_INGRESO',inplace=True,axis=1)

  return data
```

Now after all the preprocessing required for the model we have all integer values or categorical encodings that represents does original string present in the raw data.



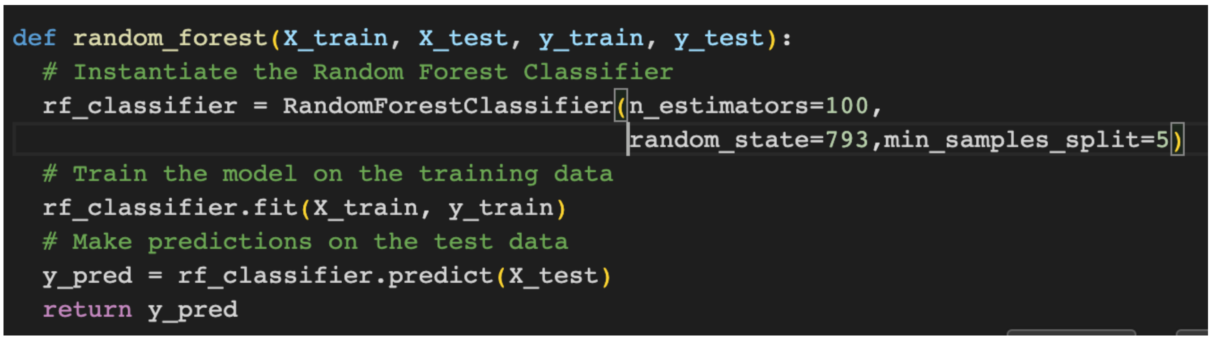
Age	Department	Education	EducationField	Gender	JobLevel	JobRole	MaritalStatus	MonthlyIncome	Ovetime	EnvironmentSatisfaction	J
17	72	21	1	1	563	80	3	1000.00	0	4	
18	2	21	1	1	388	68	3	1238.33	0	3	
18	13	21	1	1	50	70	3	1000.00	0	5	
18	33	21	1	1	610	137	3	1000.00	0	3	
18	33	21	1	1	1493	96	3	1000.00	0	5	
...	...	...	...	...	...	...	...	...	...	...	
68	19	21	1	1	552	95	3	1000.00	0	3	
68	33	21	1	1	552	95	3	1000.00	0	5	
69	56	21	1	0	407	78	3	1025.00	0	4	
69	57	21	1	1	346	204	0	1025.00	0	4	
76	0	21	1	1	216	219	0	1000.00	0	5	



Ovetime	EnvironmentSatisfaction	JobSatisfaction	PerformanceRating	WorkLifeBalance	Resigned
0	4	5	5	5	0
0	3	3	5	3	0
0	5	4	3	3	0
0	3	3	5	5	0
0	5	3	3	3	0
...	...	...	...	...	...
0	3	5	3	3	0
0	5	3	3	3	0
0	4	3	3	4	0
0	4	3	5	3	0
0	5	3	4	4	0

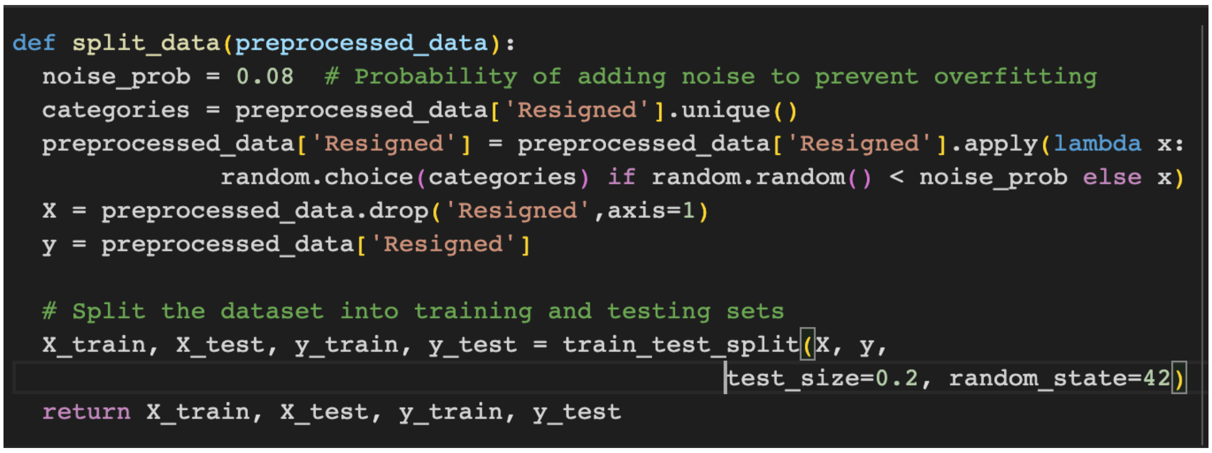
b. ML model

For the machine learning model, random forest was required which is an ensemble model and gives a decision based on the three base structure. Sklearn random forest was used with n-estimator 100, random state 793 and minimum sample 5. The **random_forest** function takes training and testing data split and fit random for its model on that data then did prediction on test data set and returns the set of predictions from the function



```
def random_forest(X_train, X_test, y_train, y_test):
  # Instantiate the Random Forest Classifier
  rf_classifier = RandomForestClassifier(n_estimators=100,
                                          random_state=793,min_samples_split=5)
  # Train the model on the training data
  rf_classifier.fit(X_train, y_train)
  # Make predictions on the test data
  y_pred = rf_classifier.predict(X_test)
  return y_pred
```

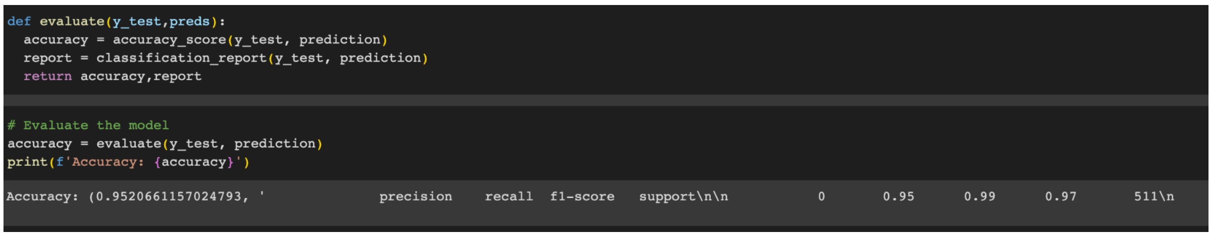
Before passing the data to the machine learning model the data needs to split into training and testing set which is done by split_data function it takes the preprocess data then add random noise to the data to prevent overfitting and then split the data into X and Y using train_test_split function it divides 20% data to the test set and remaining 80% to the training set.



```python
def split_data(preprocessed_data):
  noise_prob = 0.08  # Probability of adding noise to prevent overfitting
  categories = preprocessed_data['Resigned'].unique()
  preprocessed_data['Resigned'] = preprocessed_data['Resigned'].apply(lambda x:
              random.choice(categories) if random.random() < noise_prob else x)
  X = preprocessed_data.drop('Resigned',axis=1)
  y = preprocessed_data['Resigned']

  # Split the dataset into training and testing sets
  X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                      test_size=0.2, random_state=42)
  return X_train, X_test, y_train, y_test
```

c. Model Evaluation

At the final stage after data preprocessing and model training we need to evaluate the model performance based on the precision, recall and accuracy matrix. The train model was giving 96% accuracy, 99% precision and 97% recall which represents good evaluation against a machine learning model.



```python
def evaluate(y_test,preds):
  accuracy = accuracy_score(y_test, prediction)
  report = classification_report(y_test, prediction)
  return accuracy,report
```

```python
# Evaluate the model
accuracy = evaluate(y_test, prediction)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: (0.9520661157024793, '              precision    recall  f1-score   support\n\n           0       0.95      0.99      0.97       511\n
```

2. Backend

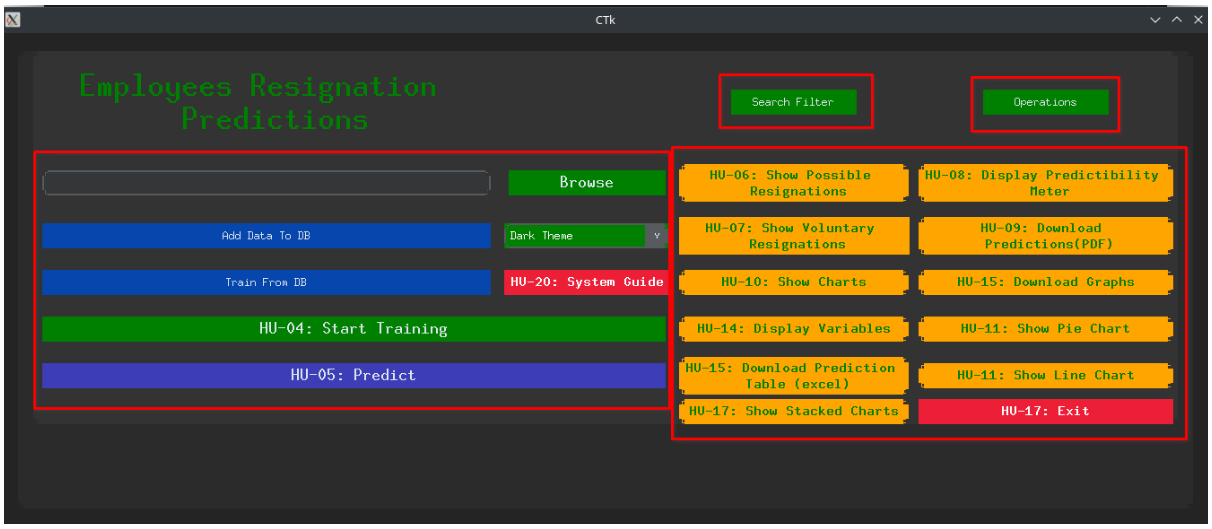
In the Backend Section, the code is divided into different parts which are then connected with the User Interface in order to provide functionalities to the buttons. The parts of backend are divided as below:

1. **GUI** : This section is the main dashboard section connecting each n every button to different tabs such as Database Operations, Search Filters, Graphs Operations etc. In order to Load the User Interface, the user has to first run this section in order to access other modules or to perform his/her desired operation. Different functions have been built and integrated according to each n every button. For Example, if a person desired to train the model from the DB, he/she would need to hit the ‘Train From DB” button. To run this specific operation, its specific function would be called which will use the object in order to access the method of ‘preprocess’ from the ‘resignationpredicton’ file.

2. **ResignationPrediction** : This section is considered to be the core of this project. Which is about to preprocess, train and predict the dataset. In order to access these processes independently, a class has been built which consists of a initializer, preprocess, train, predict and classification_report methods. When a class has been Initialised, the initializer is being called, which then creates the class variables to access the database. The prerocess method is used to process the entire dataset either from reading the excel file or database in order to make the data ready for training and divides it into train and test sets. The train method is built to train the processed dataset using the RandomForest Classifier which contains 100 estimators and min sample split of 5. Then is the predict method, which is used to measure the performance of the machine learning model. Classification report has been built alongside with the addition of a new column into '**predicted resignation**' the test set. The last part of this module is the Classification Report, even though this part and the predict method are almost quite similar. The main difference between these two is its representation. The predict method is used to measure the performance of the AI system i-e Random Forest Classifier and create and modify the test set. While on the other hand, the classification report is used to convert the information in a meaningful representation which then can be accessed via class object and can be directly integrated with the buttons.
3. **Search Filter (Popup)** : This module is used to build the search filters of the main dashboard. The search filter contains of operations such as
    a. Search by Age : Displays the filtered information by age for the DataFrame created using the predict method.
    b. Search by Gender : Displays the information from the DataFrame with respect to the gender either Male or Female
    c. Search by Predicted Resignation : Displays the information from the DataFrame with respect to the predicted value(possible resignation) either Yes or No.
    d. Search by Age Range : Displays the information from the DataFrame with respect to the Age Range. From and To Age range.
4. **DataBase** : The module is created to store the information into the database which is SQLite3. The information of workers can be either entered by the user manually using the User Interface or directly imported from the Excel sheet. This module consists of both the operations and UI code. Operations in this module are all related to database query, such as: Insert, Show Table, Delete Specific Worker record, Delete Table. For each n every operation its User Interface is built according to which mostly includes Pop Ups of independent Frames.

3. User Interface

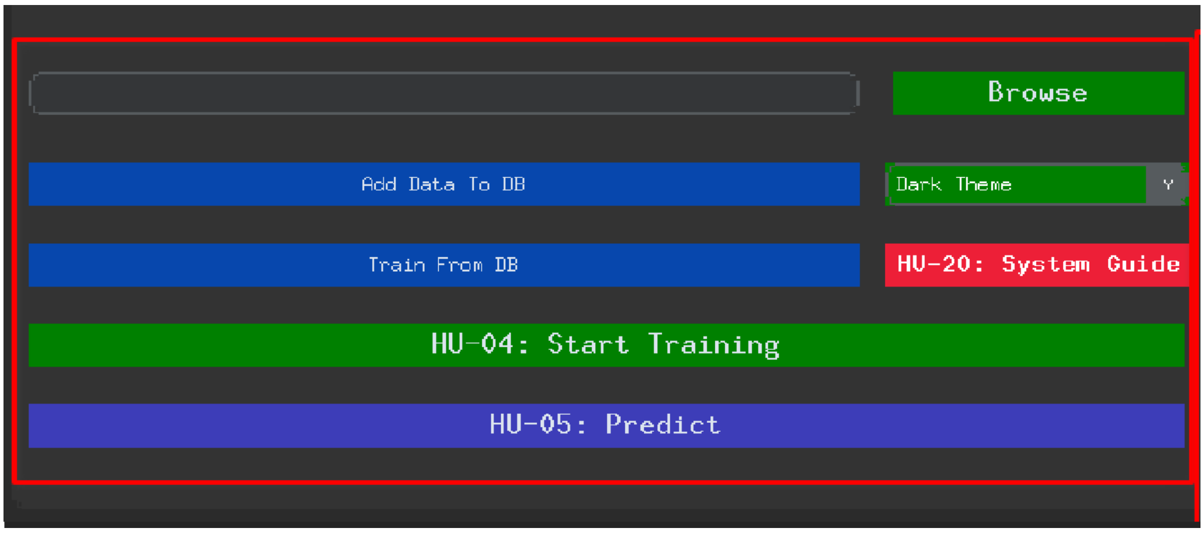
The sections of the UI is Divided into 4 main components.



From the Above Image We can see that the dashboard consists of 4 sections.

a. Data Loading
b. Data Operations
c. Data Filter
d. Analysis

<u>Data Loading</u>

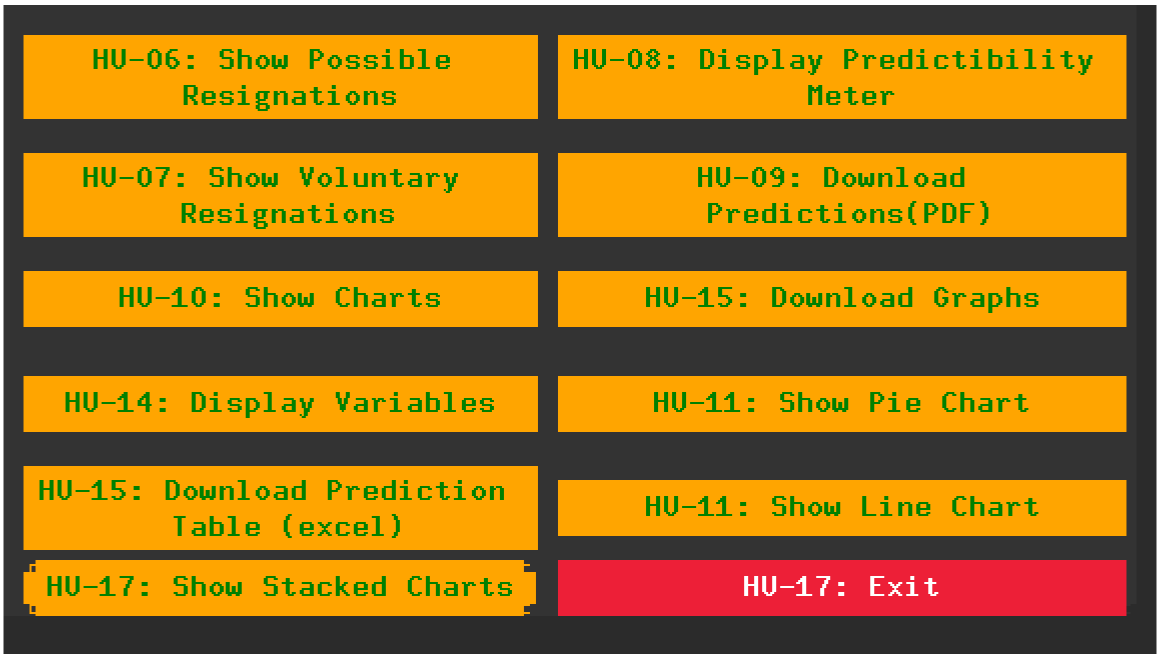


This section contains below buttons:

1. Add data to DB : To add the uploaded excel sheet to database

2. Train From DB : To train the system directly from the database
3. Start Training : To train the system from the uploaded excel sheet only
4. Predict : To predict the values on test set and measure its accuracy.

Data Analysis


HV-06: Show Possible Resignations
HV-08: Display Predictibility Meter
HV-07: Show Voluntary Resignations
HV-09: Download Predictions(PDF)
HV-10: Show Charts
HV-15: Download Graphs
HV-14: Display Variables
HV-11: Show Pie Chart
HV-15: Download Prediction Table (excel)
HV-11: Show Line Chart
HV-17: Show Stacked Charts
HV-17: Exit

In this section Different operations are being performed:

Show Possible Resignations : To show the info of all employees which have possibilities of resignation

Display Predictability Metre : To display the accuracy of the system of test set0 based on F1-Score, Accuracy, Precision and Recall.

Show Voluntary Resignation : To display the total number of Resignations

Download Predictions : To save the predicted information in PDF Table Format

Show Charts : To display the BarCharts in a single Canvas

Download Graphs : To save all possible graphs in a specific folder

Display Variables : To display all Column names and show their unique values

Show Pie Chart : To display all possible Pie Charts

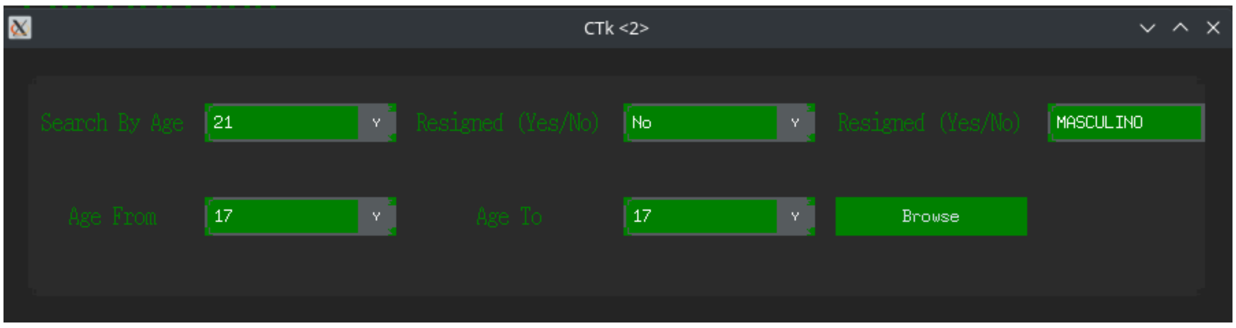
Download Prediction Table (Excel) : To Save the predictions in Excel sheet

Show Line Chart : To show all possible line charts

Show Stacked Charts : To show a stacked Chart

Exit : To get out of the system

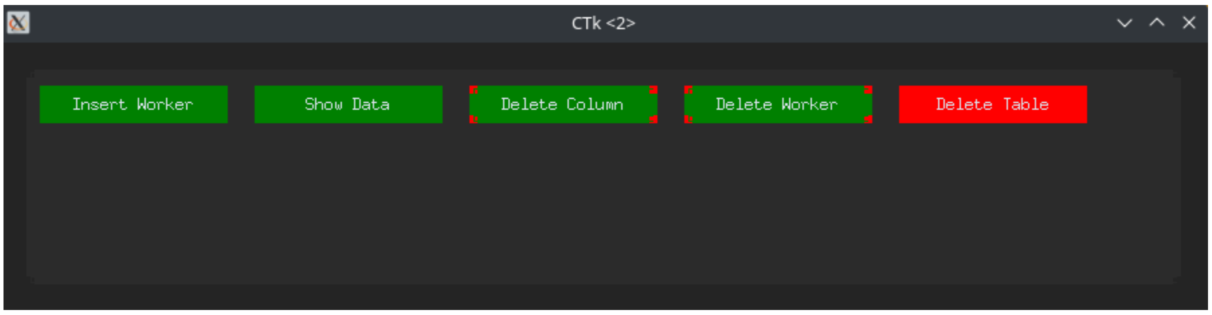
Data Filter



In the Above filter section, we can do the follow filter search from the possible resignation data.

a. Filter by Age
b. Filter by Resigned Yes/No
c. Filter By Male/Female
d. Filter by Age Range

Data Operations



In the Data Operation Sections, we have different functionalities

1. Insert Worker : To Add worker in the database
2. Show Data : To Display All Data
3. Edit Worker : To Edit the worker from DataBase with respect to the User Unique ID
4. Delete Column : To Delete any specific column
5. Data Worker : To delete a record by unique ID
6. Data Table : To delete the Table