Hackathon

FINAL ASSIGNMENT(BEEM063)

Proposal:

We, Group-5, are trying to create a personal finance application, e-CashMate, that will integrate user's bank accounts, savings and current as well as expenses from credit cards, into one place and segregate the expenses under different category heads instead of showing them all together and will show the expenses through monthly and weekly interactive graphs. Through the use of Machine Learning, it will help users to briefly understand his future spending pattern if he keeps spending at the current rate. Our product will also help users to take Investment and loan decisions based on their current spending and saving rate. It will help users to know how much loan they can afford monthly as an EMI, based on his available monthly funds. In case he has overspent, the application will help him calculate the amount of expenses he needs to reduce for the EMI to run for the targeted loan. After making EMI payments, if surplus funds are available, it will give suggestions on how to manage them. My initial function in the group is to create (i) the interactive charts of the expenditure datasets. (ii) Complete the Machine Learning model to predict future expense behaviour. (iii)Complete the monthly expense report and email notification system. I assume that I will start creating the first point using ReactJS, a special form of JavaScript and use CSS to design the website that will contain all the components. The ML part is the trickiest of all and a huge challenge that I have taken. There are several ways that this part can be done, and I have assumed that I will take a minimum of 3 to 4 weeks to learn the basics of Machine Learning and then try out all the appropriate ML models for the expense tracker. I also assume that if we fail to integrate live expense data we may need to introduce Plaid API transaction data. Another big learning area will be the application of Javascript. In my part, I feel the expense charts, alerts and notifications, future expense behaviour and the monthly and weekly graphical charts are realistically achievable. Another realistic portion I feel I can add if time permits is a voice-enabled software that will show the user only those expense categories, charts and parts of monthly reports that he asks for. In all what I am looking forward to from this project is test and improve my technical capabilities and acquire sound knowledge in programming languages. I strongly believe I can complete my part of the project within the stipulated time under the guidance of my project supervisor Jack Rogers and hope to develop my teamwork and learn from others in my group who are theoretically may have more sound knowledge than me.

Activity Log:

Week1-2: As per our proposal I started researching the kind of personal finance application that we would do. My main search criteria were based on the main components of the personal-finance application. So, I went through multiple websites of the existing personal finance apps. I made a list of the leading application names that specialize in the region. I planned to get a complete idea of the different parts of the budgeting and expense tracker. I went through each website of my list starting from Yolt, Emma, Chip, Savings Goals, Moneybox, Curve, Coconut, etc. I observed the features of the applications, their advantages and also try to locate their disadvantages. For that, I went through the user review or suggestion section in the play store. I observed what the users are expecting more on average and that most of the applications are failing to give.

Week 3: I went through with the idea of creating a front end and back end for the application. I devoted time to trying to understand the way ReactJS works and how that can be applied in creating personal finance applications. We as a group at first thought of designing the front end with CSS besides creating the application. I understood the basics and started implementing ReactJS to create budget tracker. Then I tried the use of context API and finally Material UI to design the front end. As I moved on with the front-end development, after discussing with my team, I realized that there is a roadblock to the use of Javascript particularly when we planned to use Machine Learning to showcase the future spending pattern of the user. After some extensive research, I realized that Machine Learning can only be done using python and not through JS. After a discussion with my module lead, Jack, and subject expert Sarunas, we decided that we can carry out both the front and back ends simultaneously and store them in the docker.

Week 4-5: I devoted 5 days to understand the concept of docker and how we can connect both ends using docker. For that, I started going through the concept of docker-compose. As I researched further, I realized that I have to go through the concept of NoSQL Databases. At this point, I realised that I have spent more than the days I have allotted for this part and after a discussion with my group members, I dropped the idea of doing a front end. From this point, I concentrated totally doing everything on the backend using python. I have noted the different components that our application should have like a login section, account integration section, category-wise spending segregation, weekly and monthly spending charts, alert creation, machine learning to track future expense behaviour. I found out that for our purpose we can import the Plaid Transaction API into our jupyter notebook. Once my groupmate had imported the plaid API data, I started working on the modified data for breaking and grouping the transaction details under different categories like food and drinks, shop, recreation, etc and represent them as a pie chart using matplotlib package and the weekly and monthly expenses in the form of bar diagram. The category of Plaid expenses had one single big payment part. So, I discussed with my teammate and broke the payments in the dataframe into smaller divisions like insurance, bill, and home rent payment. So that the monthly bar diagram chart looks more realistic.

Week 6-7: For these two weeks my plan was to go through the different approaches that I can take to build the ML and expense alert or notification part for our application. I decided to undergo the future expenditure analysis using the Backward Elimination process. After understanding its theory, I started on the coding part. I made use of the sklearn package for breaking my dataset into training and testing sets and then conducted the linear regression using the statmodel.api from the sklearn package. Once the regression analysis of the two sets was done applying the NumPy package, I went on to do the backward elimination of the test set. Once I had the datapoints after backward elimination, I made them into a dataframe, so that I can create a graphical representation of the real points and the predicted future expenditure points for the future days. Then I went ahead with my plan to create email notification of expenditure spending alerts. My idea was that that user will get a monthly spending chart, where the user can set a limit to his different spending categories, and at the month-end, he will receive notification on where he has overspent. For example, if he has set his food limit at 100 pounds per month and his expenses on food becomes 110 pounds, it will show him a notification that his food and drinks expenses are more than his food limit. Due to some error in the codes, although I figured out how to attach an email notification, I had to settle with a notification that the user will see only through the application and not separately through email.

Week 8,9 and 10: I plan to first work with my team to include our unique idea of a loan application based on the current spending pattern. I completed my task of converting my groupmate’s mathematical formula into a python function. The basic package that I used for building the model is NumPy financial. According to the variables expenses that we took, I made a pie chart on the expenses, net savings, and emergency fund, where the whole pie represented the maximum affordable loan amount. Most of the last 2 weeks went on troubleshooting the codes to answer the three main questions that I have spoken about in my proposal above. One example of the difficulty that I faced was linking the Plaid API values with our monthly variables. We took Plaid data for two years but realized that it had 22 months of data instead of 24, so the variable expenditure data we have taken for the loan application section needs to be divided by 22 to get monthly data and make them par with the Plaid API data.

Reflection

This application was to the point of our proposal. As a group, we made sure that we will stick to the main aspects that we have mentioned in our proposal, and we have successfully managed to do that till the completion of our prototype. But there’s no denying that there is room for improvement on different parts of the application. My part went through several ups and downs starting with integrating multiple bank accounts from different banks of the user into the application. I tried to integrate one of our accounts from multiple banks to integrate but due to some errors in the code that I used, it didn’t materialize. But I am satisfied with the way I came up with the idea of integrating Plaid transaction API. Although the data was random, still it served our purpose. Another achievement on my part was the way I pulled out the Machine Learning part. Starting from having no knowledge of it to learning it and then finally implementing was a huge challenge. We discussed that I have to complete this and I am satisfied with the way I exceeded my own expectation regarding the challenge. The result of the ML using Backward Elimination would have given a much more improved future predicted points if we had taken more monthly data points, like 5 years or 60 months of historical data from the Plaid API. More past data points will impact the training and testing sets and hence would have given more accurate results. Given the time constraint, I could not complete the Markov Chain Model or the Weighted Arithmetic Mean, or the combination of both approaches to automating future expenses. I strongly believe that this is a point where any potential investor can compare the Backward Elimination approach with the above approaches and see which one is giving a more convincing result. At the beginning of the application, as discussed, I tried doing the login registration page. For that, I tried using the Flask package to create a simple login page. Although I made the login credentials, the codes did not support me as I had not made any database for storing user detail that can be pulled in while executing the python codes. At this point, I realized that weeks 3,4 and part of 5 did not go completely my way. I tried hard on finding a way to connect the front end containing the CSS file, the ReactJS portion with the back end using the docker container. But due to limited knowledge on my part, I was forced to drop out of the front-end portion and carry the entire application building using python and hence compromise on the designing aspect of the application. For further improvement, I believe one can use PHP for interactive charts and user interface and back that up with MySQL database for storing details of the customers, expenses, income, and balance so that the login application page can work so that whenever user logs in using his email id and password, he will be able to see his entire financial details. In that section only I had plans of adding Speechly, a voice-enabled software. This will help the user to record transaction details put them into their respective transaction category heads, just by listening to the user’s voice and the person doesn’t need to type the transactions. Coming on to the notifications part, my primary aim was to add email notifications of monthly expense reports along with normal notifications inside the application. So, this part went partially according to my plan due to some code errors that did not happen. Every time I tried to link the alert reports with a working email account, an error seemed to occur. I believe in the real-life scenario when the application will process live transaction data, once the user sets a limit to each category, at the end of the month, an email containing the full monthly report will reach the user and he doesn't need to open the application separately. This can also be improved by automating the limit using ML, where the application will suggest the user certain limits based on previous expenditures habits. But I believe manual limits, in this case, would be better. The last part, which is the loan application section, went beyond our team's expectations. It was the first time that I worked with the NumPy Financial Package in python. In most cases, we can find the loan calculation based on current savings on separate websites. We as a group feel that we have exceeded our expectations in bringing them under the single roof of e-CashMate. A big challenge that I overcame in this section is connecting the Plaid API data with the variable expenditures that we have assumed. One of the biggest lessons that I learned from this section is that writing financial equations in python is far more difficult than it seems. Working with the financial equation one important thing that I learnt while coding the equation and debugging the errors is working with maximum functions. Since I was working on a financial model, the values of all the equations should be either 0 or positive value. Only the savings value can be negative. But for simplicity as my groupmate has assumed that it will not be negative. While creating the maximum affordable loan pie chart, keeping income, expenditure, and savings all together was hard. Whenever a value of an equation became negative, the pie chart percentage portion showed error. So, the last three weeks were quite hard because Python is casesensitive, and I had to debug the errors to execute the correct equations. A potential improvement in this section is to add more realistic factors that can impact investment and loan decisions like age and gender. I believe that the equations that we have put in seem to be working in general. The investment and loan decisions equations will vary from age to age and from different gender. So, I believe the loan equations can be made for different age groups both for males and females. So that way the application can be more user specific. An up-gradation on this part would be whenever a user registers with the application, the gender and age are noted and when the person moves into the investment decision section, he gets a more accurate investment decision advice more related to him or her. For future improvement, investors can add features like personal loan applications, add credit scores, and options of monitoring them. The successful use of Dash app to visually represent the pie charts, bar diagrams and monthly expense reports went beyond my expectation. It was on me as my idea of creating the front end with the website designing did not go ahead, I was all along searching for an alternative. After some hard research, I realized the dash app can solve our problem as it is very similar to a front-end website development using ReactJS. I am very happy with the way my groupmates grabbed my idea and successfully launched the website running. It was a good lesson learnt as to the use of plotly and flask to create the dash app and I was satisfied with the way I was able to help my group stick to the way we had earlier planned to finish our application. It can be further improved in the future by adding tags and notes to transactions, show whether any of the transactions are recurring or not, an option of adding or removing them from the analytical display. Add quarterly and semi-annual expense charts along with weekly and monthly.

In the 10 weeks that I have devoted to building the application, it acted as an icebreaker for me. I feel much more confident in coding and other technical parts. The best skill I learnt was the use of Machine Learning. There were hours when I felt frustrated not being able to crack the codes to execute the function, but I didn’t disrupt my focus and that paid off. I played around with all the three models that I mentioned of the ML application and although I used the Backward Elimination process, I came very close to use the combination of MCM and the Weighted Arithmetic Model. I believe now I can use these models or create a new one for future projects. I also went through a rigorous analysis of ReactJS and learnt quite a lot on how to efficiently work on website and designing using CSS. As a group, this application went quite well leaving out that docker container part joining front and back end, which I feel is quite significant for any future investor to take up to take it to next level. Many of its parts are automated or have the potential of future automation to make the application faster and more user specific. So as a group I feel we have been able to keep our word in making something unique by bringing together a personal finance application, future expenditure behaviour, and loan and investment decision based on current savings as well as predicted future savings. None of the existing applications has been able to integrate all these features under one roof all connected in a more advanced way and that is what makes our application e-CashMate unique and appealing.