Xtreme Programming Exercise

Initial Description from Client:

- Suppose our client is a multinational company providing various IT-related services. Those services include but are not limited to, cloud services (storage services, packages, computation services), etc.
- The company is receiving an enormous number of emails from their customers. Those emails are related to various subjects which again include, but are not limited to, request for refunds, service quality-related issues, some suggestions, etc.
- Each of these emails can be classified into different classes on more than one different level. For example, an email can be classified as 'complaint' for level 1, 'IT service' for level 2, 'cloud autoscaling' for level 3, and 'under-provisioning of HDD' for level 4, where each level can have multiple classes to classify in.

The objective of the Project:

Our client has hired hundreds of agents to go through the emails and respond to queries of their customers. As a result, our client is bearing the high cost to provide this service to their customers. Also, their customers' experience is very bad because they must wait for days (even weeks) to get a response from the agents.

Our client has decided to develop a chatbot that will automatically read the emails of customers and respond to them in the best manner. As a first step towards developing such a chatbot, our client wants to develop an ***email classifier that can classify emails into multiple classes*** (one class on each level).

Sample Data Format:

Ticket id	Interactio	Interactio	Mailbox	Ticket Sur	Interactio	Innso TYP	Type 1	Type 2	Type 3	Type 4
24310	68365		support.e	[Company	Mxxxxx@	AppGaller	AppGaller	Others		
24075	67730		support.p	[AppGalle	Beschreib	AppGaller	AppGaller	Problem/I	AppGaller	Can't update Apps
24075	67762		support.p	Re: RE : [A	Danke Na	AppGaller	AppGaller	Problem/I	AppGaller	Can't update Apps
23748	66706		support.p	[AppGalle	Descriptic	AppGaller	AppGaller	Suggestio	AppGaller	Others
23740	66684		support.eu	@Service	Ho pagato	AppGaller	AppGaller	Problem/I	Third Part	Refund
23683	66429		support.e	Service Su	Product: A	AppGaller	AppGaller	Suggestio	VIP / Offe	Offers / Vouchers / Promotions
23683	67433		support.e	Re: RE : Se	Hello , go	AppGaller	AppGaller	Suggestio	VIP / Offe	Offers / Vouchers / Promotions
23683	67516		support.e	Re: RE : Se	I have to a	AppGaller	AppGaller	Suggestio	VIP / Offe	Offers / Vouchers / Promotions
23683	67915		support.e	Re: RE : Se	Sure, the	AppGaller	AppGaller	Suggestio	VIP / Offe	Offers / Vouchers / Promotions
23624	66266		support.p	[GameCer	Descriptic	AppGaller	AppGaller	Problem/I	AppGaller	Can't download Apps
23555	65977		support.p	[AppGalle	Descrição	AppGaller	AppGaller	Problem/I	General	Cannot connect - Server
23534	65883		support.p	[AppGalle	Beschreib	AppGaller	AppGaller	Problem/I	AppGallery	-Install/U Can't install Apps
23477	65661		support.p	Re: suppo	Huawi hea	AppGaller	AppGaller	Others		
23452	65616		support.p	[AppGalle	Descriptic	AppGaller	AppGaller	Problem/I	AppGaller	Can't install Apps
23429	65569		support.p	[AppGalle	Descriptic	AppGaller	AppGaller	Problem/I	AppGaller	Can't install Apps
23395	65482		support.p	[AppGalle	Descriptic	AppGaller	AppGaller	Others		
23230	65024		support.p	[AppGalle	Descriptic	AppGaller	AppGaller	Others		
23229	65022		support.p	[AppGalle	Описание	AppGaller	AppGaller	Others		
23212	64982		support.p	[AppGalle	OK Описа	AppGaller	AppGaller	Others		
23194	64946		support.p	[AppGalle	Beskrivels	AppGaller	AppGaller	Problem/I	AppGaller	Other download/install/update issue
23193	64944		support.p	[AppGalle	Beskrivels	AppGaller	AppGaller	Problem/I	AppGaller	Other download/install/update issue
23189	64938		support.p	[AppGalle	Описание	AppGaller	AppGaller	Suggestio	General	Personal data

Our client provided some data to further understand their technical requirements (They promised to provide more data next week and will keep providing it during the project duration). The data is attached to Moodle. At this stage, the client only provided the data for a single business scope i.e., the emails they got about their application gallery service. The format of the sample data provided is given above. For more detail see the worksheet attached on Moodle page.

Initially planned activities:

Seq.	Activity to perform	The expected way to perform	Activity Resource	Estimated Duration
1	Data Selection	Skipping few columns	Deleting from worksheet	3 hour
2	Data grouping	Ticket based grouping of conversation	Data group-by facility in python	3 hour
3	Deal with multiple languages	Translate them all into English	May be using some NLP resources using Python's libraries	2 days
4	Deal with noises in data	E.g., removing frequent less important words (thank you for your email)	Regular expressions in python	1 days
5	Dealing with multi-level data	Multi-level modelling or multi-class	Brainstorming activity: Language do have support for both	1 day
6	Textual data representation change	Numeric representation of text	Libraries in python	1 day
7	Dealing imbalanced data	E.g., similar number of minimum records for each class	Libraries in python	Half day
8	Decide on we want supervised or un-supervised learning: Brainstorming activity			
9	Data preparation for modelling	Separate training testing data	Libraries in python e.g., train_test split in sklearn	2 hour
10	Model selection for email classification	Use an appropriate (SOTA) model that can be use textual data for classification purpose	E.g., Random forest implementation in Python	4 hour
11	Model training, and testing for email classification	Input training data, train model, then use trained model for testing on test data	E.g., Random forest implementation in Python	3 days

Tasks to do:

Except activity 5, you have been provided the solution for all of the activities presented above. Most of the solutions are in the form of source code but for activity 8 we don't need any source code instead we decided to use supervised learning approach. Why?

Your today's task as an AI Engineer is to adapt the Xtreme Programming Agile method practices for our client's AI project for email classification.

1. **Planning game** – determine the scope of the next release by combining business priorities and technical estimates.

   We want to release an applicable/working release and we need to decide what activities-related code we should release. The decision is yours!

   List the activities code you will release and give a reason to release that.

2. **Small releases** – put a simple system into production, then release new versions in a very short cycle.

   At this stage, we will assume the code is in running form as deployed into the production system of a company (we will change this assumption in the next labs). You may revisit the activities you selected in practice 1 (planning game) and may make your system more simple if possible (e.g., by reducing some activities).

3. **Metaphor** – all development is guided by a simple shared story of how the whole system works

   A very simple but comprehensive story that you and your teammate agree with. To develop such a story, the initial description of the project from the client and the project objective may be helpful (see the top of this file).

4. **Simple design** – system is designed as simply as possible (extra complexity removed as soon as found)

   You may draw some use cases of the system to share with each other.

5. **Testing** – programmers continuously write unit tests; customers write tests for features

   Suppose you are a programmer and I am the customer. My first test case is your code should show me what is the probabilities of each email to be in each class (see display result solution).

6. **Refactoring** – programmers continuously restructure the system without changing its behavior to remove duplication and simplify

   Now suppose that I was your old colleague and I code all of the activities given in the table above. That code snippets are available to you. You need to put all those snippets together and refactor the code in order to execute the code. Note: I am no more accessible as I left the organization.

7. **Pair-programming --** all production code is written with two programmers on one machine

   The executable code should be a collective effort of you and your team member.

8. **Collective ownership** – anyone can change any code anywhere in the system at any time

   Somehow divide the work and let your team member do approximately half of the work and the remaining half work you should be doing. For example, if you selected a total of 6 activities, you may work on the first 3 activities and your partner on the second 3.

9. **Continuous integration** – integrate and build the system many times a day – every time a task is completed

   Integrate the work of each other and produce a fully executable solution

10. **40-hour week** – work no more than 40 hours a week as a rule

    You promise to never work more than 40 hours. If you have done that much this week inform your colleague that you will start working next week.

11. **On-site customer** – a user is on the team and available full-time to answer questions

    I am here to answer your question as a user of the system

12. **Coding standards** – programmers write all code in accordance with rules emphasizing communication through the code