***ML Phising Classifier***

1. **Problem Statement**

* To build classification model to predict whether a website is phising website or not on the basis of given set of features.
* Companies/Government want to protect the people from such websites and prevent the fraudulent activities to happem again

1. **Description of data**

* Dataset contains continuous values encoded as 0 or 1.
* Columns inside the dataset are:

1. having\_IP\_Address: -1, 1 if domain has ip address as domain name than phisiing else not
2. URL\_Length: if the length of url is very long or contains redirection information if length < 54 legitimate or > 54 phising
3. Shortining\_Service: if tinyurl present -> phising else not (eg.bit.ly/)
4. having\_At\_Symbol: if url has @ then phising else not
5. double\_slash\_redirecting: if // is > 7 then phising else not
6. Prefix\_Suffix: domain name includes prefix symbol then phising else not
7. having\_Sub\_Domain: dot in domain part=1 then legitimate if =2 suspicious else > 2 phising
8. SSLfinal\_State: https and age of certificate > 1 – legitimate, if https and certificate not trusted – suspicious else phising
9. Domain\_registeration\_length: domain registration < 1 then phishing else not
10. Favicon: favicon (image) loaded from external domain – phising else not
11. Port: if non standard port number used then phishing else not (eg. 21, 22, 23, 80, 443, 445, 1433, 1521, 3306 are standard)
12. HTTPS\_token: http token used then phishing else not
13. Request\_URL: % of request url < 22% 🡪 legitimate, >= 22 and 61% 🡪 suspicious else phishing
14. URL\_of\_Anchor: long url of the anchor tag
15. Links\_in\_tags: different links in the different tags
16. SFH: Server form handler is blank then phishing
17. Submitting\_to\_email: directly use the mailto function
18. Abnormal\_URL: hostname not included in url
19. Redirect: more redirection then phishing
20. on\_mouseover: statusbar changes after hovering
21. RightClick: rightclick is disabled
22. popUpWidnow: lots of pop up windows with text field
23. Iframe: lot of iframe borders than phishing
24. age\_of\_domain: new domain tend to be phishing
25. DNSRecord: no dns record then phishing
26. web\_traffic: traffic on the website , check the rank of the websites >300000 then phishing
27. Page\_Rank: on which page the website occurs
28. Google\_Index: whether the website is index or not
29. Links\_pointing\_to\_page: less links point to the fake websites
30. Statistical\_report: statistical reports published by some companies check if the website there
31. Result: based on this the website is phishing or not
32. **Application architecture and module division**

* Bigger problem: How to classify the data to decide whether website is phishing or not?
* Better to break down the development into small parts so that changes to be made in end of module doesn’t affect the other modules. Multiple members part of the project so better to divide the modules
* Broker into small subgroups:

1. How to read the data
2. How to validate the data
3. How to do data preprocessing and how to train a model on the data
4. How to do hyperparameter tuning for the model

* **Step 1**: Data ingestion

1. Data for training - client provides or stores the data needed at a particular location, aggregate multiple data sources
2. Data validation – discuss with client the datatype of variables, number of variables, whether any columns contain only null values
3. Data transformation – missing values conversion to null, categorical values in commas or “” and maybe not accepted in DB
4. Data insertion in DB – after transformation insert the data inside the database for further development

* **Step 2**: Training Pipeline / Step

1. Export the data in CSV from DB and csv acts as train data
2. Data preprocessing – perform EDA, check if there are any null values present, convert categorical values into numerical values, if data is imbalanced or normalized
3. Data clustering - to increase the accuracy of the model we divide the data into individual clusters and build model for each cluster separately
4. Hyperparameter tuning - to increase the performance of the individual model selected for each cluster
5. Model saving – save the model for each cluster individually

* **Step 3:** Deploy on cloud

1. Create metadata for pushing the app onto the cloud server
2. Start and test the application

* **Step 4**: Prediction Pipeline

1. Data validation – discuss with client the datatype of variables, number of variables, whether any columns contain only null values
2. Data transformation – missing values conversion to null, categorical values in commas or “” and maybe not accepted in DB
3. Data insertion in DB – after transformation insert the data inside the database for further development
4. Export the data in CSV from DB and csv acts as train data
5. Data preprocessing – perform EDA, check if there are any null values present, convert categorical values into numerical values, if data is imbalanced or normalized
6. Data clustering - to increase the accuracy of the model we divide the data into individual clusters and build model for each cluster separately
7. Call the model for specific cluster number stored
8. Make prediction and export the prediction in a csv file

* **Step 5:** Model retraining

1. When new patterns detected these changes must be aggregated to the model.
2. Provide the prediction + train data to the model for retraining
3. Logging and monitoring framework
4. **Code**:

Main .py -🡪 1. Validation step – read data, validation, transformation, insert into DB, export

to csv file.

* 2. Training – read train data, data preprocessing, data clustering, model finding,

Model tuning, deployment

* 3. Prediction – validation,

Prediction – model saved loaded into memory and make predictions

* Data for training: User provides different training batch files
* Synchronize logging, asynchronous – code doesn’t wait for completing the logging faster and individual
* Data Validation: Whether the data sent by the client is valid or not as per requirement given. File name is correct or not based on agreement, if we reject the data push into bad folder else put in good data folder. We use the schema files created. We delete the good and bad directory as the good data is stored inside the database. Check the length of the timestamp given in the file name format, check the number of columns present in the data, check if any column contains just all column values as null values.
* Data Transformation: DB doesn’t accept Nan value so we transform to NULL using fillna. Add quotes to all values present in the categorical values for insertion into the DB.
* Data Insertion in DB: Put the good raw data inside the DB using the datatypes given inside the schema. Export the good raw data table into csv.
* Data preprocessing: Use the exported csv file as input. Perform EDA to understand the data and what processing is needed for the data.

Strip or remove the white spaces in categorical values.

Replace the “?” with NaN in the dataset and check number of missing values.

No null values present inside the dataset so no imputation necessary

Check the distribution for everycolumn present boxplot, violin plot, and check if data is balanced or not.

* Model Training: Fetch the data and perform the preprocessing decided after doing the EDA. Separate the target and features. Perform clustering on the cleaned data and create clusters. We can apply cluster specific algorithms for better performance. Loop individual clusters and apply different models. Perform standard scaler on the input data.

Logistic regression, DT, RF, SVM, NB, KNN, XGBoost algorithms can be used for classification.

* We perform hyperparameter tuning for the selected algorithms and check the performance of the model. Then save the model for further prediction. We used SVM and XGBoostClassifier. If same label classes present then we use the accuracy score else we can use the roc\_auc\_score for the check. Save the model for each cluster created.
* Prediction –

1. perform validation – for filename, no of columns, all of null values
2. perform transformation – replace nan, insert into db, export csv as input
3. data preprocessing – missing values and imputation, std == 0 dropped, drop unnecessary columns
4. Perform clustering to determine which cluster it belongs to using Kmeans.predict
5. Then based on cluster assigned use the respective model for each individual cluster. Reassign the encode class category back on the predicted file.

* Index.html – default for every browser it call webage it returns httpget

Render template to display respective html pages.

* Deployment for cloud 🡪

1. Requirements.txt – import of packages are included here, as cloud needs instruction for cloud deployment (pip freeze requirements.txt)
2. Azure deployment stes:

Create account and create a web app

Go to deployment center, choose git and choose service

Use url