Configuration Manual

MSc Internship
Msc. CyberSecurity

Chirag Sharma
Student ID: X18151485

School of Computing
National College of Ireland

Supervisor: Dr. Muhammad Iqbal

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Chirag Sharma		
Student ID:	X18151485		
Programme:	MSc. Cybersecurity	**Year:**	2019-2020
Module:	Cybersecurity		
Lecturer:	Dr. Muhammad iqbal		
Submission Due Date:	8 Jan 2020		
Project Title:	Feed Forward MLP SPAM domain Detection Using Authoritative DNS Records and Email Log		
Word Count:	**800 words**		
Page Count:			

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

I agree to an electronic copy of my thesis being made publicly available on NORMA the National College of Ireland's Institutional Repository for consultation.

Signature:

Date: 8 Jan 2020

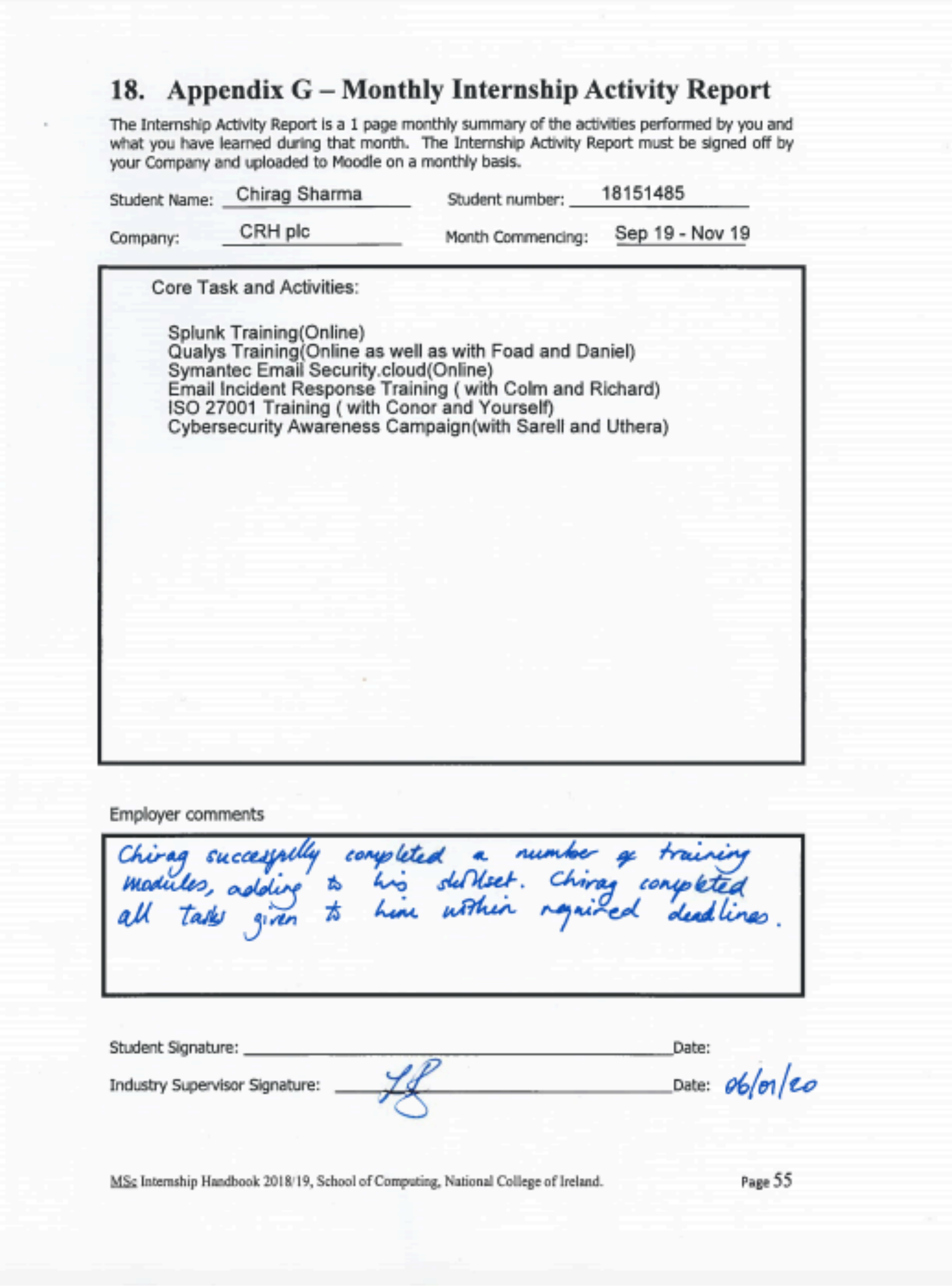
PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Internship Report



18. Appendix G – Monthly Internship Activity Report

The Internship Activity Report is a 1 page monthly summary of the activities performed by you and what you have learned during that month. The Internship Activity Report must be signed off by your Company and uploaded to Moodle on a monthly basis.

Student Name: Chirag Sharma Student number: 18151485

Company: CRH plc Month Commencing: Sep 19 - Nov 19

Core Task and Activities:

Splunk Training(Online)
Qualys Training(Online as well as with Foad and Daniel)
Symantec Email Security.cloud(Online)
Email Incident Response Training (with Colm and Richard)
ISO 27001 Training (with Conor and Yourself)
Cybersecurity Awareness Campaign(with Sarell and Uthera)

Employer comments

Chirag successfully completed a number of training modules, adding to his skillset. Chirag completed all tasks given to him within required deadlines.

Student Signature: ______________________ Date:

Industry Supervisor Signature: ______________________ Date: 06/01/20

MSc Internship Handbook 2018/19, School of Computing, National College of Ireland. Page 55

Configuration Manual

Chirag Sharma
Student ID: x18151485

1 Introduction

This document provides a walkthrough of the implementation with technologies used to build our detection model in a stepwise manner. We start off by collecting the email data from spam archive and then cleaning it to fetch domain name and other feature and then querying to DNSBLs and fetching active DNS records from Avro. Since email headers can be complicated to be read or retrieved manually from email files , email parser module makes things easy to retrieve email headers in a dictionary format. For DNSBLs there are SPAMHAUS, SURBL open blacklists and 'spam-lists' library has built-in functions to lookup for spam domains. For our Neural network classifier, we first baselined with Logistic Regression and then use first the generic MLP and after that we try improve the accuracy of our result using Gradient descent algorithm, Regularization factor and hidden layers.

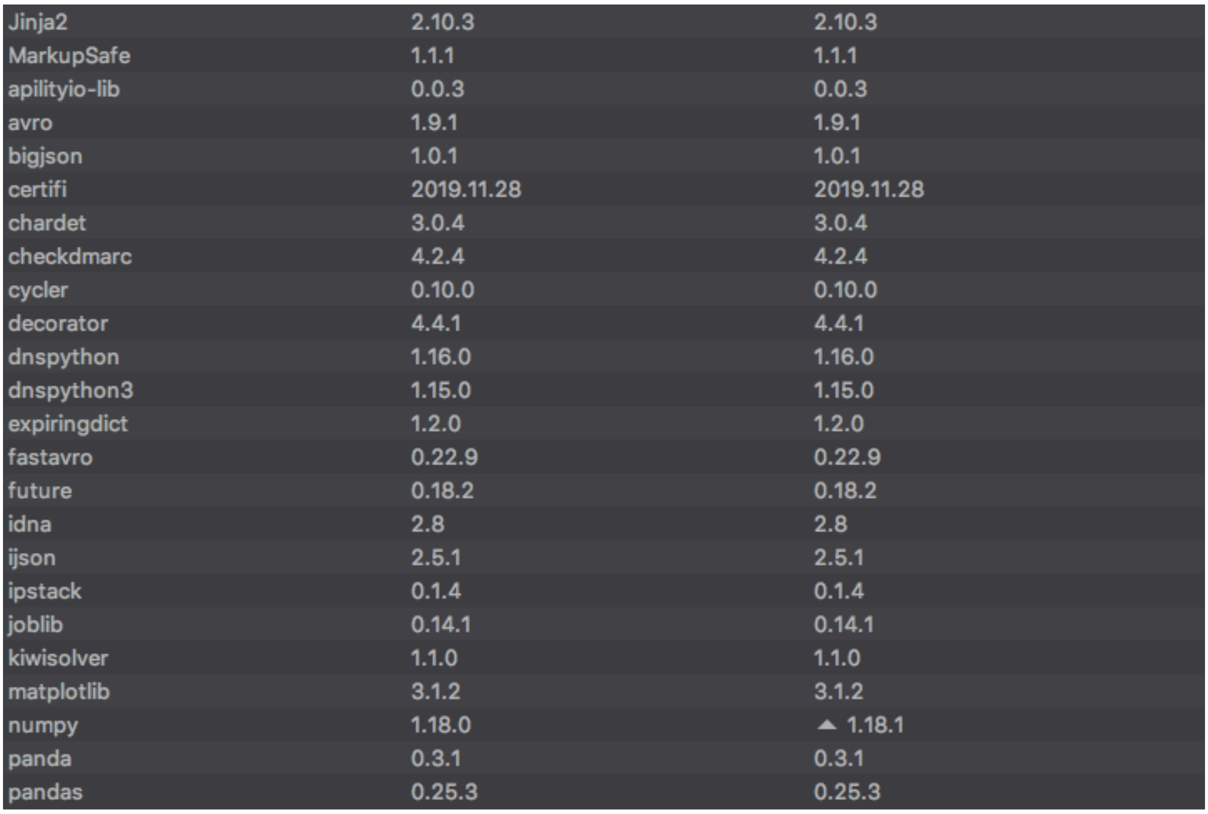
2 System Configuration

- OS – MAC OS High Sierra
- Processor – Intel Core i5 – 2 Cores
- 64-bit version
- Ram - 8 GB
- Python version – 3.7
- Pip 2.1.3
- Pycharm -2019 -3.1

3 Walkthrough

3.1 Setting up Environment

We setup a virtual environment using Virtualenv which loads all the necessary libraries and freezes the environment for particular usage. We install pip which is a package manager for python. Also setting up Pycharm with the virtualenv interpreter.

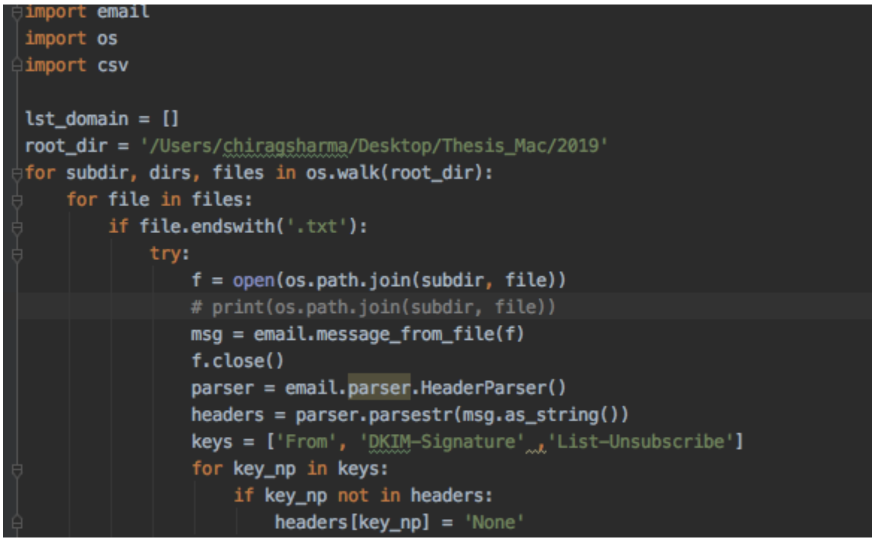


Jinja2	2.10.3	2.10.3
MarkupSafe	1.1.1	1.1.1
apilityio-lib	0.0.3	0.0.3
avro	1.9.1	1.9.1
bigjson	1.0.1	1.0.1
certifi	2019.11.28	2019.11.28
chardet	3.0.4	3.0.4
checkdmarc	4.2.4	4.2.4
cycler	0.10.0	0.10.0
decorator	4.4.1	4.4.1
dnspython	1.16.0	1.16.0
dnspython3	1.15.0	1.15.0
expiringdict	1.2.0	1.2.0
fastavro	0.22.9	0.22.9
future	0.18.2	0.18.2
idna	2.8	2.8
ijson	2.5.1	2.5.1
ipstack	0.1.4	0.1.4
joblib	0.14.1	0.14.1
kiwisolver	1.1.0	1.1.0
matplotlib	3.1.2	3.1.2
numpy	1.18.0	▲ 1.18.1
panda	0.3.1	0.3.1
pandas	0.25.3	0.25.3

Figure 1: Packages Installed

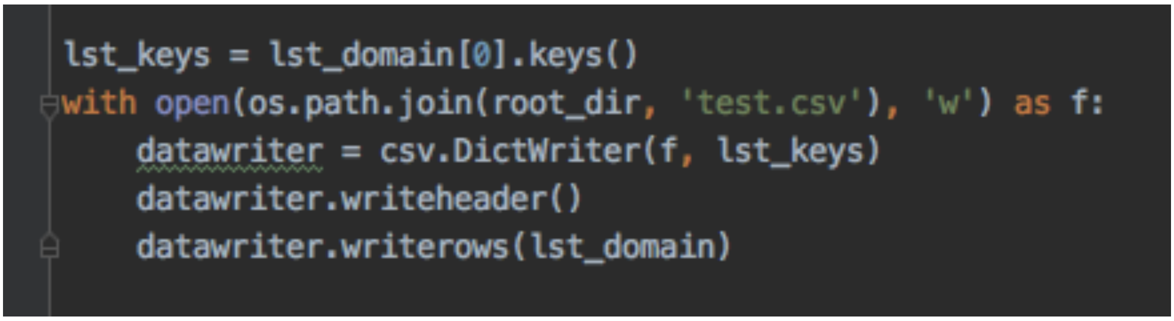
3.2 Email header extraction

For this process we require email library and csv library to read email message from text, email has a function named Header parser which converts all the parts of email into a key value pair in the form of a dictionary and that dictionary can be written into csv using Dict reader module.



```
import email
import os
import csv

lst_domain = []
root_dir = '/Users/chiragsharma/Desktop/Thesis_Mac/2019'
for subdir, dirs, files in os.walk(root_dir):
    for file in files:
        if file.endswith('.txt'):
            try:
                f = open(os.path.join(subdir, file))
                # print(os.path.join(subdir, file))
                msg = email.message_from_file(f)
                f.close()
                parser = email.parser.HeaderParser()
                headers = parser.parsestr(msg.as_string())
                keys = ['From', 'DKIM-Signature','List-Unsubscribe']
                for key_np in keys:
                    if key_np not in headers:
                        headers[key_np] = 'None'
```

Figure 2: Email Header Parsing



```
lst_keys = lst_domain[0].keys()
with open(os.path.join(root_dir, 'test.csv'), 'w') as f:
    datawriter = csv.DictWriter(f, lst_keys)
    datawriter.writeheader()
    datawriter.writerows(lst_domain)
```

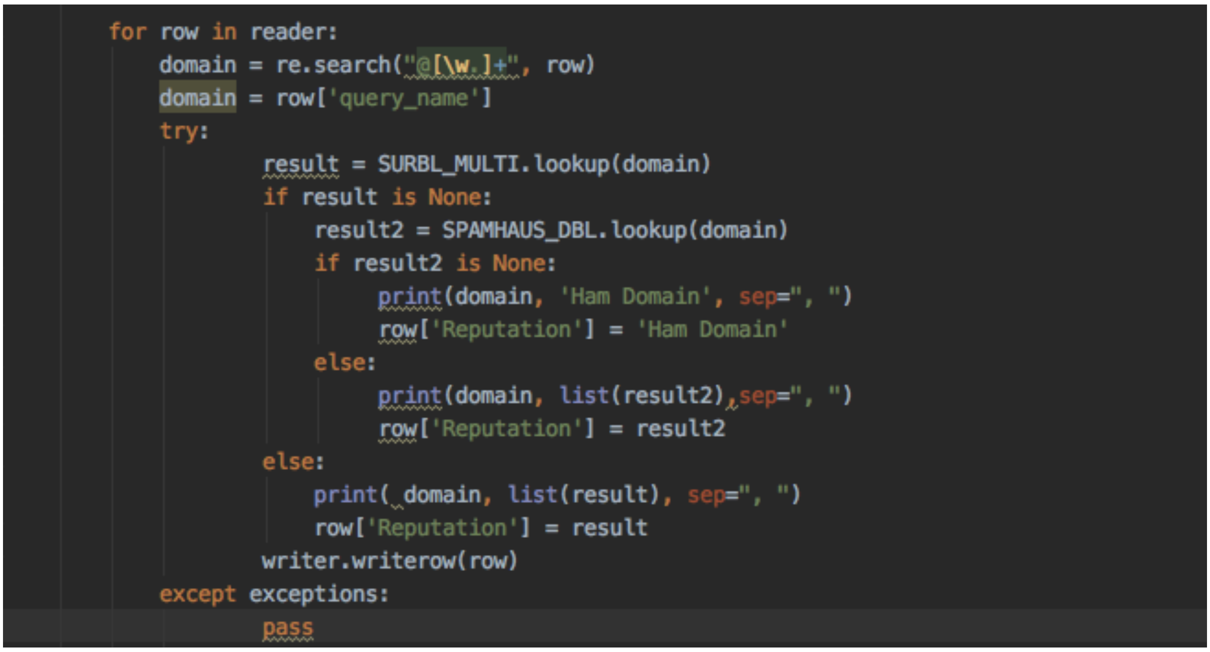
Figure 3: Convert email header dictionary to CSV

3.3 Cleaning Email headers and Query DNSBL

Using spam-lists package we query onto SPAMHAUS and DNS BL but before that we find out the regex pattern for email in ‘From’ column as they are fetched in format as below.

john.doe@example.com" <john.doe@example.com>

Hence, we fetch the domain name and the recursively query in nested IF because to check both of the DNSBL first we query SPAMHAUS and if the result is none from SPAMHAUS [2] then we query to SURBL and even from SURBL the result is none we mark the domain as a ham domain. File name:email_clean_csv.py



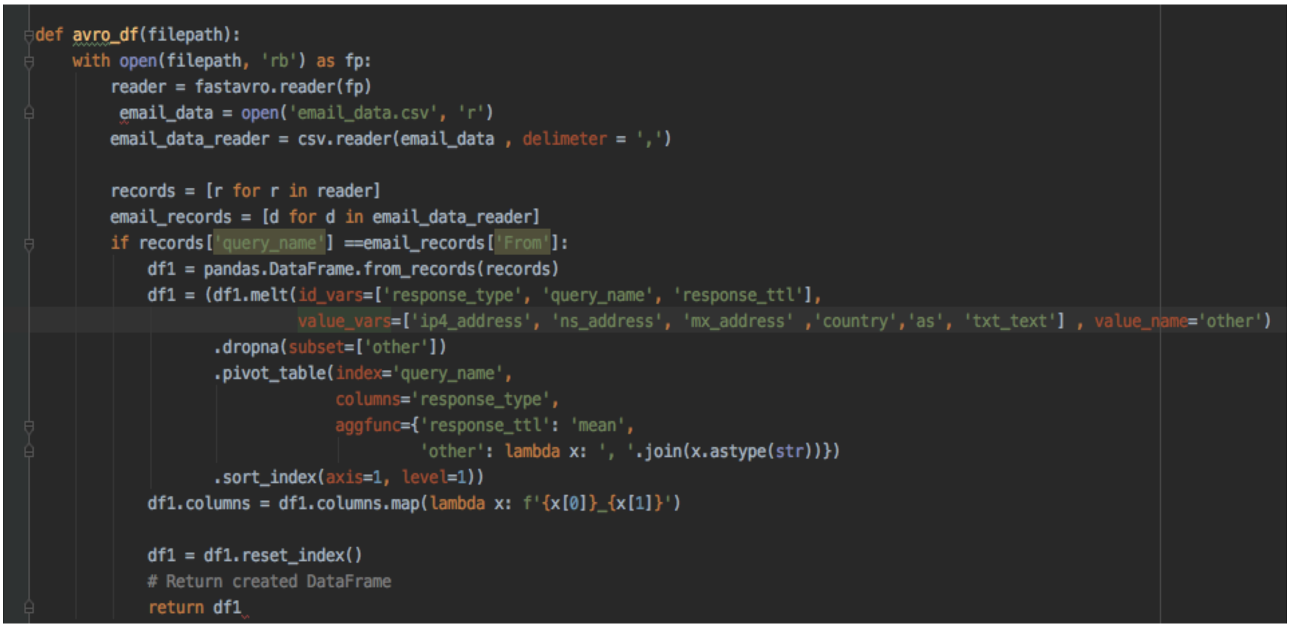
```
for row in reader:
    domain = re.search("@[\w.]+", row)
    domain = row['query_name']
    try:
            result = SURBL_MULTI.lookup(domain)
            if result is None:
                result2 = SPAMHAUS_DBL.lookup(domain)
                if result2 is None:
                    print(domain, 'Ham Domain', sep=", ")
                    row['Reputation'] = 'Ham Domain'
                else:
                    print(domain, list(result2),sep=", ")
                    row['Reputation'] = result2
            else:
                print( domain, list(result), sep=", ")
                row['Reputation'] = result
            writer.writerow(row)
    except exceptions:
            pass
```

Figure 4: DNSBL Query

3.4 Query Active DNS records

DNS records are in AVRO format and we use fastavro function to fetch IP addresses of ‘A’. ‘TXT’ and TTL values of ‘A’ and ‘TXT’, ‘NS’, ‘MX’ if the domain name matches from the ‘FROM’ field of DNS records. File name : avro_read.py

We load avro [1] records into pandas data frame and since one single DNS records consisted of 114 key value pairs and most of them were null. We used Pandas melt and pivot function to convert rows into columns and to remove null and redundant key value pairs and dumping the records on to csv file.

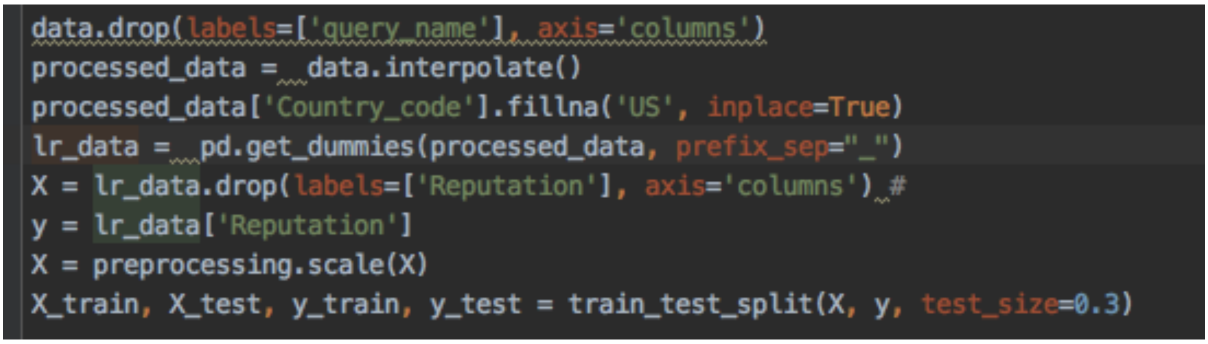


```
def avro_df(filepath):
    with open(filepath, 'rb') as fp:
        reader = fastavro.reader(fp)
         email_data = open('email_data.csv', 'r')
        email_data_reader = csv.reader(email_data , delimeter = ',')

        records = [r for r in reader]
        email_records = [d for d in email_data_reader]
        if records['query_name'] ==email_records['From']:
            df1 = pandas.DataFrame.from_records(records)
            df1 = (df1.melt(id_vars=['response_type', 'query_name', 'response_ttl'],
                                value_vars=['ip4_address', 'ns_address', 'mx_address' ,'country','as', 'txt_text'] , value_name='other')
                   .dropna(subset=['other'])
                   .pivot_table(index='query_name',
                                columns='response_type',
                                aggfunc={'response_ttl': 'mean',
                                         'other': lambda x: ', '.join(x.astype(str))})
                   .sort_index(axis=1, level=1))
            df1.columns = df1.columns.map(lambda x: f'{x[0]}_{x[1]}')

            df1 = df1.reset_index()
            # Return created DataFrame
            return df1
```

Figure 5: Fetch DNS records from Avro file

3.5 Data Exploration

Firstly, we drop the field ‘query_name’ as it consists of domain name and it is a unique identifier. After that we interpolate the data for missing values. And fill country code which had missing values with value US because most of the country codes were from US.
Next in order to convert categorical data into hot codes, we use get_dummies function of panda’s library.
Then we build the training set imto 60:30 split and map the data with training X and Y coordinates.



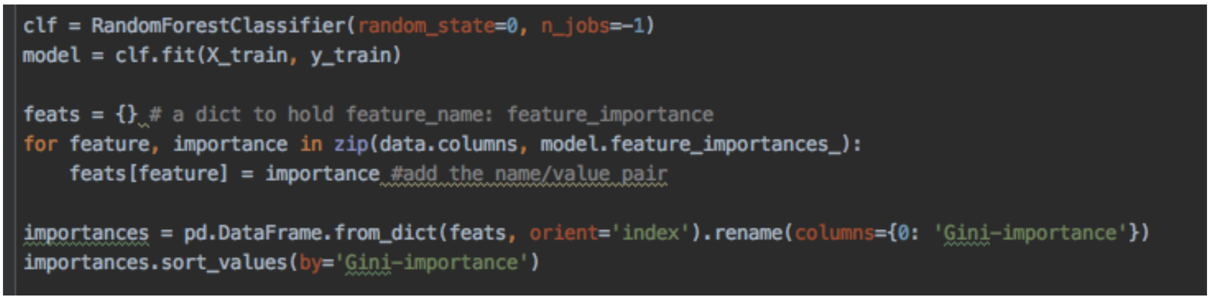
```
data.drop(labels=['query_name'], axis='columns')
processed_data =  data.interpolate()
processed_data['Country_code'].fillna('US', inplace=True)
lr_data =  pd.get_dummies(processed_data, prefix_sep="_")
X = lr_data.drop(labels=['Reputation'], axis='columns') #
y = lr_data['Reputation']
X = preprocessing.scale(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

Figure 6: Data Exploration

Now we check the correlation of all the features using corr() function of pandas library and then plot a matrix using Seaborn heatmap function.

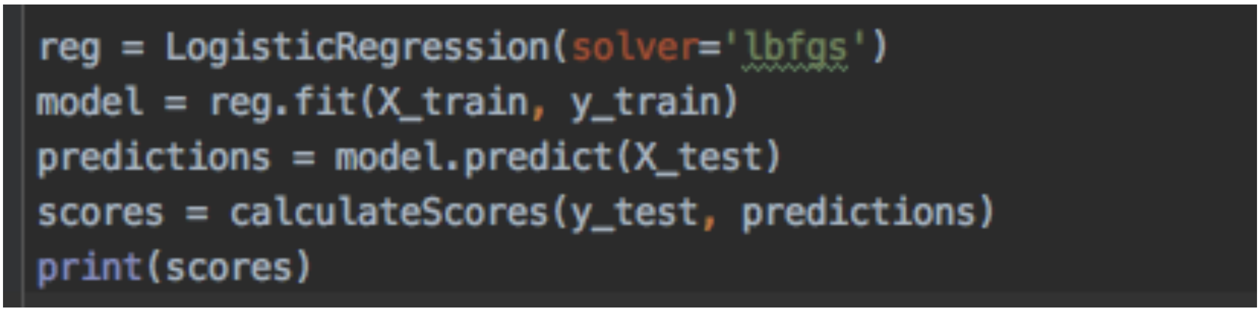
3.6 Feature importance calculation

Used the random forest classifier for calculation of feature importance and plot a bar chart using Matlibplot library. We fetch the features from dataframe store them in a dict and mapped using importances() function within random forest classifier gainst the importance scores.



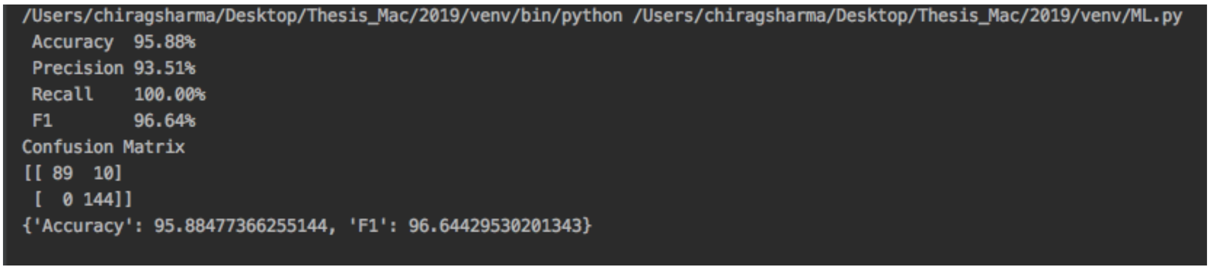
```
clf = RandomForestClassifier(random_state=0, n_jobs=-1)
model = clf.fit(X_train, y_train)

feats = {} # a dict to hold feature_name: feature_importance
for feature, importance in zip(data.columns, model.feature_importances_):
    feats[feature] = importance #add the name/value pair

importances = pd.DataFrame.from_dict(feats, orient='index').rename(columns={0: 'Gini-importance'})
importances.sort_values(by='Gini-importance')
```

3.7 Logistic Regression classifier



```
reg = LogisticRegression(solver='lbfgs')
model = reg.fit(X_train, y_train)
predictions = model.predict(X_test)
scores = calculateScores(y_test, predictions)
print(scores)
```

Figure 7: Logistic Regression Classifier

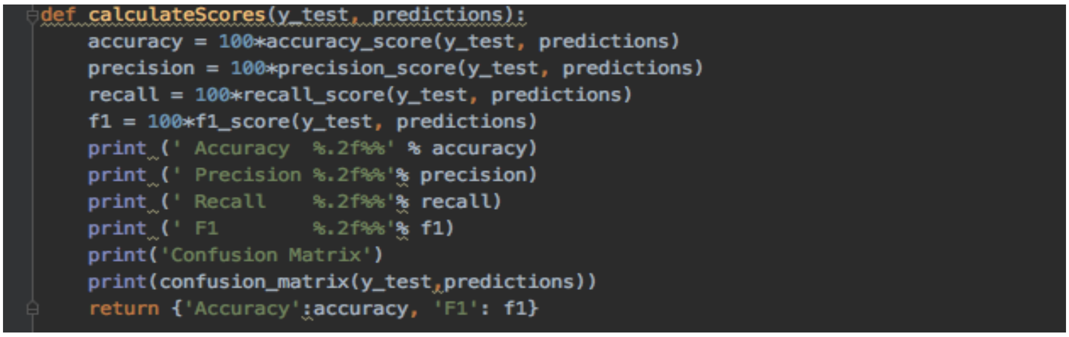


```
/Users/chiragsharma/Desktop/Thesis_Mac/2019/venv/bin/python /Users/chiragsharma/Desktop/Thesis_Mac/2019/venv/ML.py
 Accuracy  95.88%
 Precision 93.51%
 Recall    100.00%
 F1        96.64%
Confusion Matrix
[[ 89  10]
 [  0 144]]
{'Accuracy': 95.88477366255144, 'F1': 96.64429530201343}
```

Figure 8: LR Classifier Results

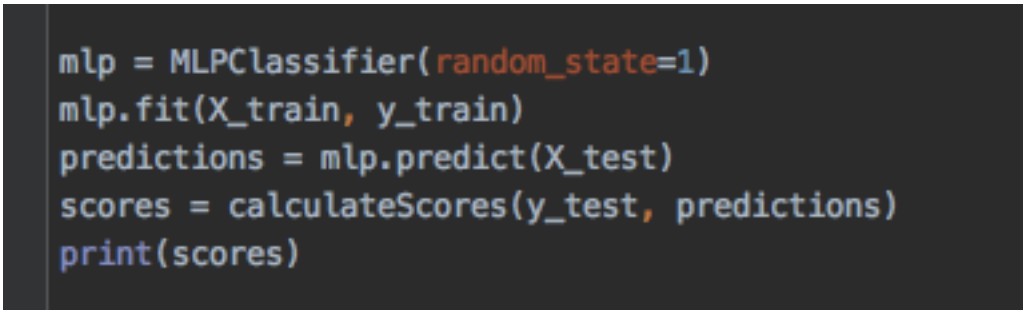
3.8 MLP Classifier

First the generic classifier without solvers . First performing the model fitting and then predict results through calculate score function which calculates Accuracy, Precision, recall and f1 score.



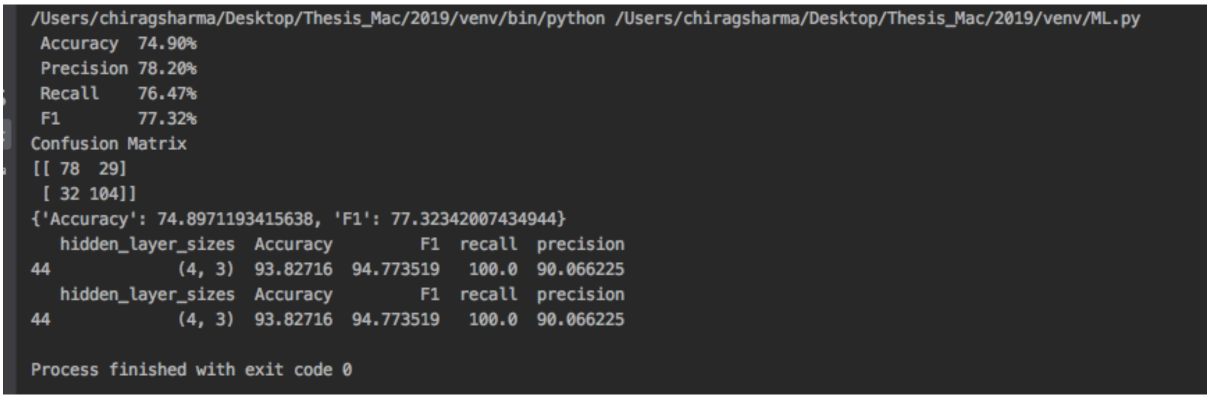
```
def calculateScores(y_test, predictions):
    accuracy = 100*accuracy_score(y_test, predictions)
    precision = 100*precision_score(y_test, predictions)
    recall = 100*recall_score(y_test, predictions)
    f1 = 100*f1_score(y_test, predictions)
    print (' Accuracy  %.2f%%' % accuracy)
    print (' Precision %.2f%%'% precision)
    print (' Recall    %.2f%%'% recall)
    print (' F1        %.2f%%'% f1)
    print('Confusion Matrix')
    print(confusion_matrix(y_test,predictions))
    return {'Accuracy':accuracy, 'F1': f1}
```

Figure 9: Calculate scores function



```
mlp = MLPClassifier(random_state=1)
mlp.fit(X_train, y_train)
predictions = mlp.predict(X_test)
scores = calculateScores(y_test, predictions)
print(scores)
```

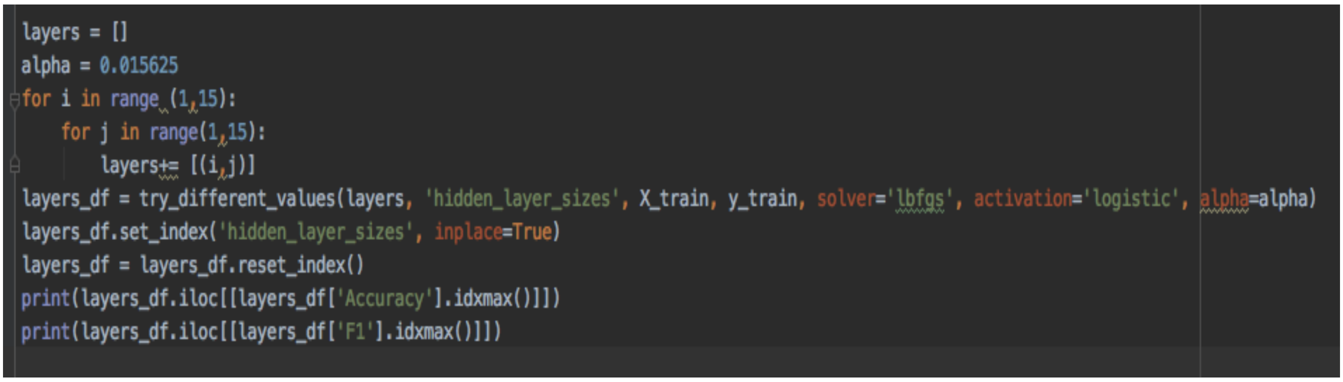
Figure 10: Generic MLP



```
/Users/chiragsharma/Desktop/Thesis_Mac/2019/venv/bin/python /Users/chiragsharma/Desktop/Thesis_Mac/2019/venv/ML.py
 Accuracy  74.90%
 Precision 78.20%
 Recall    76.47%
 F1        77.32%
Confusion Matrix
[[ 78  29]
 [ 32 104]]
{'Accuracy': 74.8971193415638, 'F1': 77.32342007434944}
   hidden_layer_sizes  Accuracy         F1  recall  precision
44             (4, 3)  93.82716  94.773519   100.0  90.066225
   hidden_layer_sizes  Accuracy         F1  recall  precision
44             (4, 3)  93.82716  94.773519   100.0  90.066225

Process finished with exit code 0
```

Figure 11: Generic MLP results

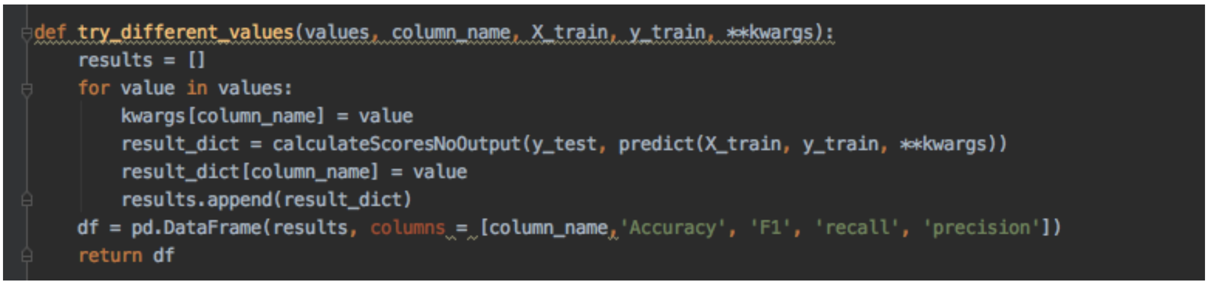
3.9 Feed Forward classifier with logistic activation, regularization coefficient and hidden layers



```
layers = []
alpha = 0.015625
for i in range (1,15):
    for j in range(1,15):
        layers+= [(i,j)]
layers_df = try_different_values(layers, 'hidden_layer_sizes', X_train, y_train, solver='lbfgs', activation='logistic', alpha=alpha)
layers_df.set_index('hidden_layer_sizes', inplace=True)
layers_df = layers_df.reset_index()
print(layers_df.iloc[[layers_df['Accuracy'].idxmax()]])
print(layers_df.iloc[[layers_df['F1'].idxmax()]])
```

Figure 12: Feed Forward MLP classifier

The above code represents feed forward classifier which takes the value of alpha and along with doubly nested loop of range 1,15 which indicates number hidden layer size combinations to try and give out the best result by passing arguments to a function name try_different values which ultimately calls Calculate scores function() to calculate the final scores.



```
def try_different_values(values, column_name, X_train, y_train, **kwargs):
    results = []
    for value in values:
        kwargs[column_name] = value
        result_dict = calculateScoresNoOutput(y_test, predict(X_train, y_train, **kwargs))
        result_dict[column_name] = value
        results.append(result_dict)
    df = pd.DataFrame(results, columns = [column_name,'Accuracy', 'F1', 'recall', 'precision'])
    return df
```

Figure 13: Try different values function

4. References

[1] Fastavro, “fastavro/fastavro,” GitHub, 20-Dec-2019. [Online]. Available: https://github.com/fastavro/fastavro. [Accessed: 08-Jan-2020].
[2]DBL - The Spamhaus Project. [Online]. Available: https://www.spamhaus.org/dbl/. [Accessed: 08-Jan-2020].