CRM DATA CLEANING

BERKADIA®

Sukrit Sen

Sana Kaur

Abhishek Anney

Bhuvana Challagalla

BERKADIA®

Leader In the Commercial Real Estate Industry

Delivers comprehensive solutions for an entire life cycle of their clients assets.

Gary Mumford

VP of Data Governance at Berkadia

Rohit Aggarwal

Associate professor, University of Utah

BUSINESS PROBLEM:

- Slow and time consuming
- Increased Error chances
- Inefficient and inaccurate
- Data Analysis is not leveraged completely on the data-set.

STRATEGIC LIMITATIONS:

- Rapid decay of CRM data captured on Salesforce.
- Frequent change in Customer Information (2% per month & 25% per year).
- Effective utilization of Sales and Marketing Teams.

OBJECTIVE OF THE PROJECT

- Usage of Quick and Efficient Machine Learning techniques.
- Identifying Key Metrics.
- Development of Robust De-deduplication Machine Learning Algorithm.

TECHNICAL REQUIREMENTS:

- Python (Programming Language)
- Pycharm as IDE.
- Pandas (Data Preprocessing)
- Fuzzywuzzy (Percentage based Matching)
- Scikit-learn (Machine Learning Algorithms)
- Numpy (Numerical Calculations)

KEY METRICS

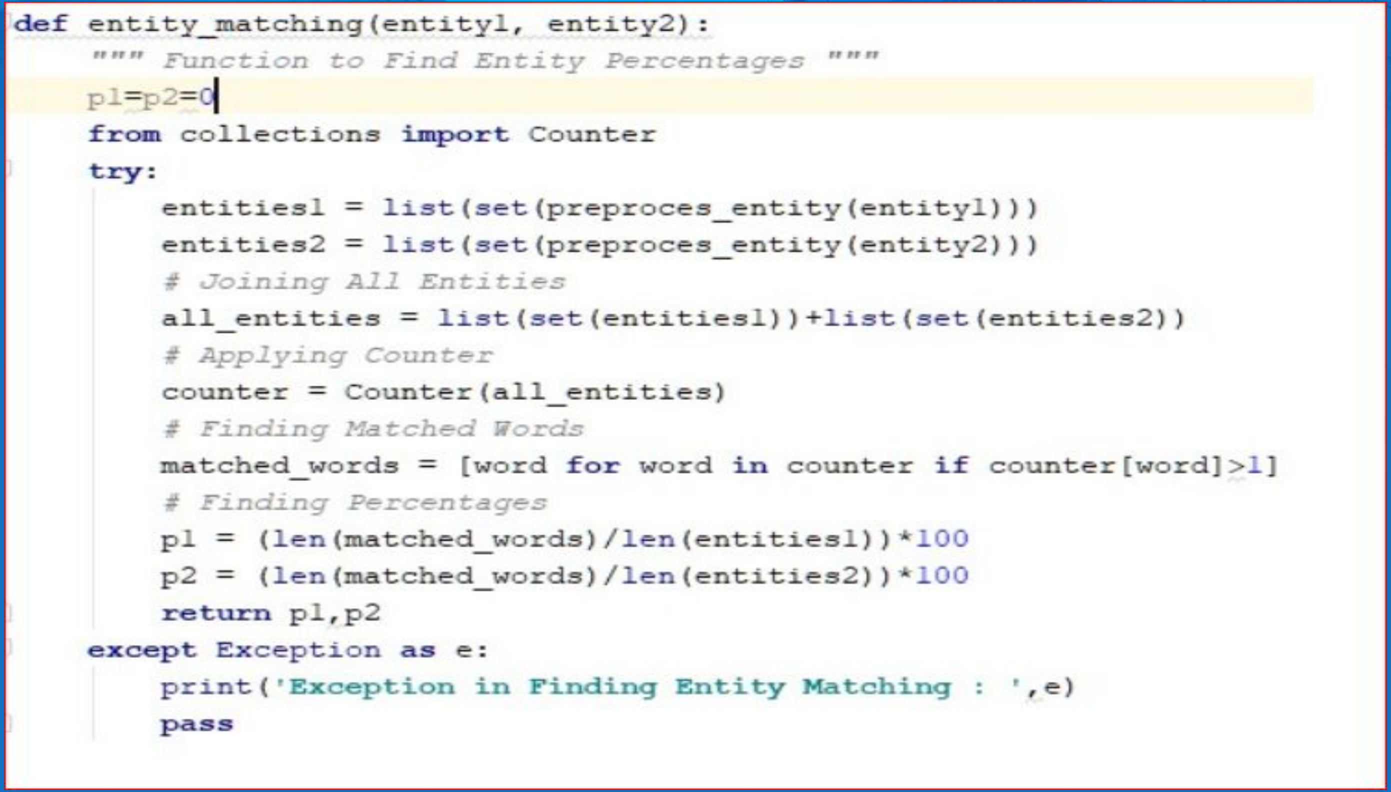
- Name (First Name and Last Name).
- Email.
- Phone.
- Title (Designation of the account owner).
- Address (Combination of Street, City, State and Country Address).

PHASE 1: PATTERN MATCHING

Character Matching	Fuzzy Token Matching

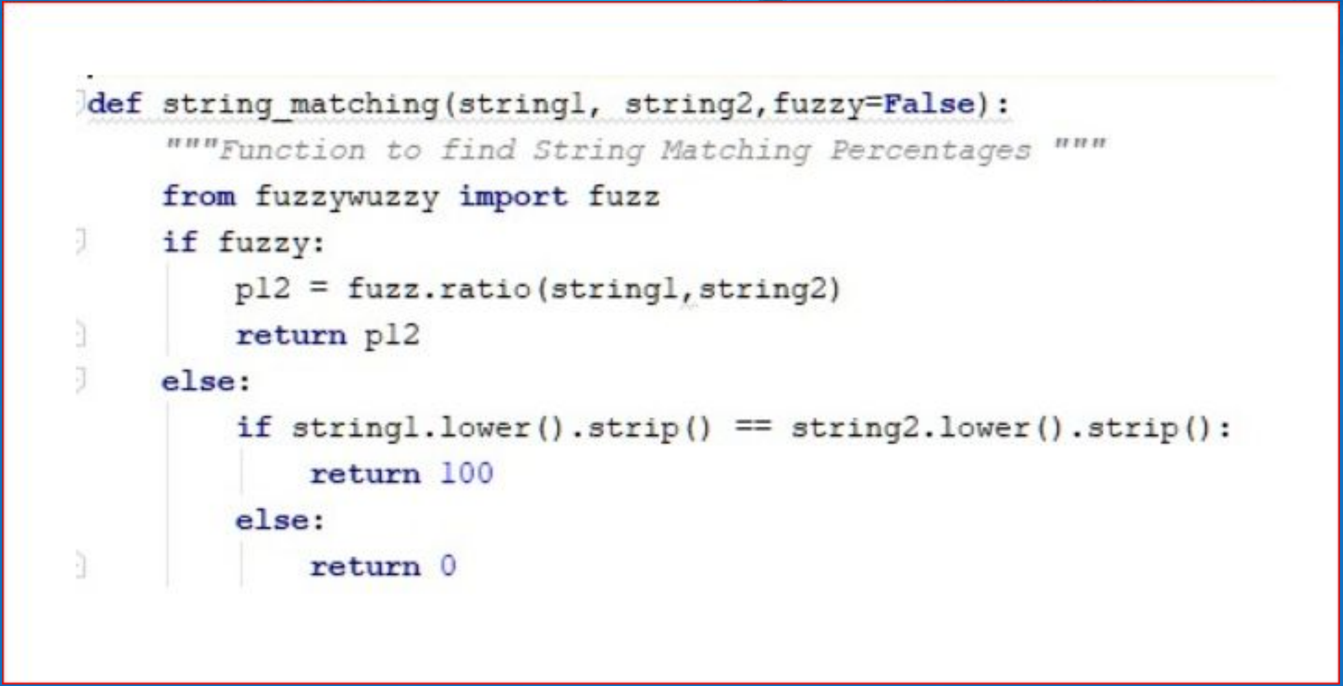
Name_Field1 = “John Doe”
Name_Field2 = “John Doe”
Matching percentages = 100%
EMAIL , PHONE Number
	Name_Field1 = [‘John’, ‘Doe’, ‘Hill’]
	Name_Field2 = [‘John’, ‘Doe’]
	MATCH= [‘John’,‘Doe’]
	Matching Percentages :
	NameField1_Matching = 66%
	NameField2_Matching = 100%
	NAME , TITLE , ADDRESS

ENTITY MATCHING :



```
def entity_matching(entity1, entity2):
    """ Function to Find Entity Percentages """
    p1=p2=0
    from collections import Counter
    try:
        entities1 = list(set(preproces_entity(entity1)))
        entities2 = list(set(preproces_entity(entity2)))
        # Joining All Entities
        all_entities = list(set(entities1))+list(set(entities2))
        # Applying Counter
        counter = Counter(all_entities)
        # Finding Matched Words
        matched_words = [word for word in counter if counter[word]>1]
        # Finding Percentages
        p1 = (len(matched_words)/len(entities1))*100
        p2 = (len(matched_words)/len(entities2))*100
        return p1,p2
    except Exception as e:
        print('Exception in Finding Entity Matching : ',e)
        pass
```

STRING MATCHING:

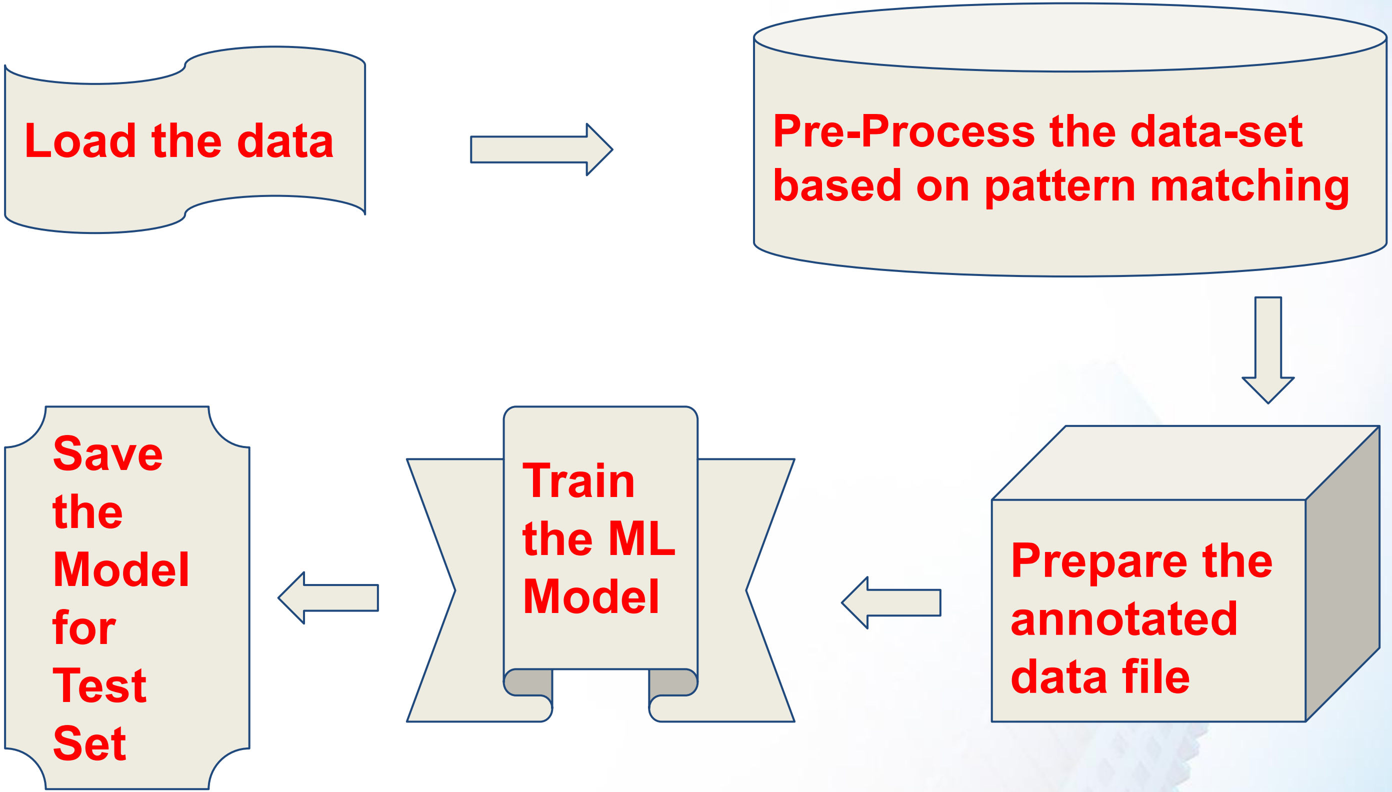


```python
def string_matching(string1, string2,fuzzy=False):
    """Function to find String Matching Percentages """
    from fuzzywuzzy import fuzz
    if fuzzy:
        p12 = fuzz.ratio(string1,string2)
        return p12
    else:
        if string1.lower().strip() == string2.lower().strip():
            return 100
        else:
            return 0
```

PROCESS FLOW:

- Percentages → Numpy Array of features.
- Data Annotation : Labeled data for Machine Learning Algorithm training.
- Train Test Split.
- Predicting and Validating Test Values.
- Saving Models for Future Predictions.

PHASE 2: TRAINING



Load the data

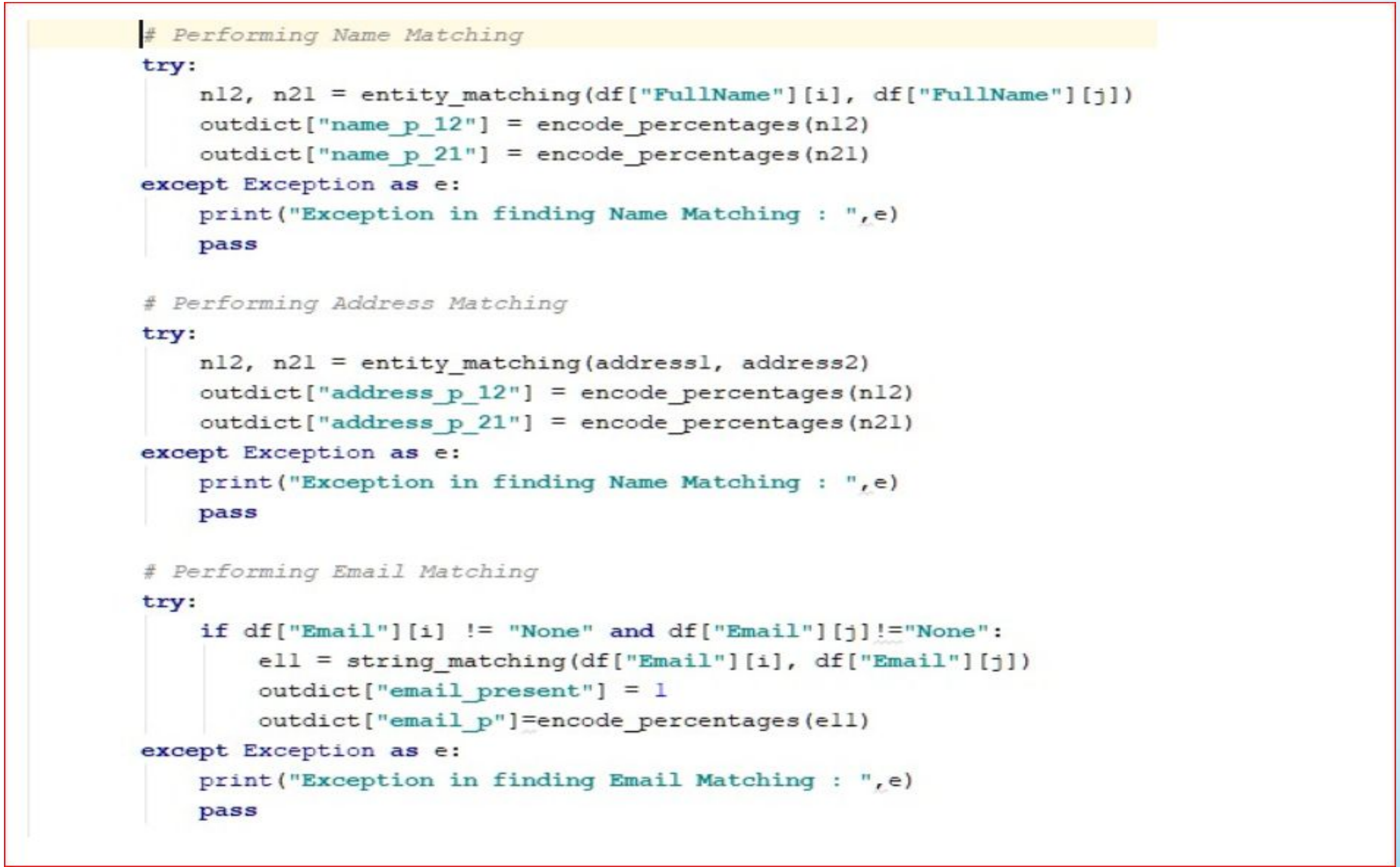
Pre-Process the data-set based on pattern matching

Prepare the annotated data file

Train the ML Model

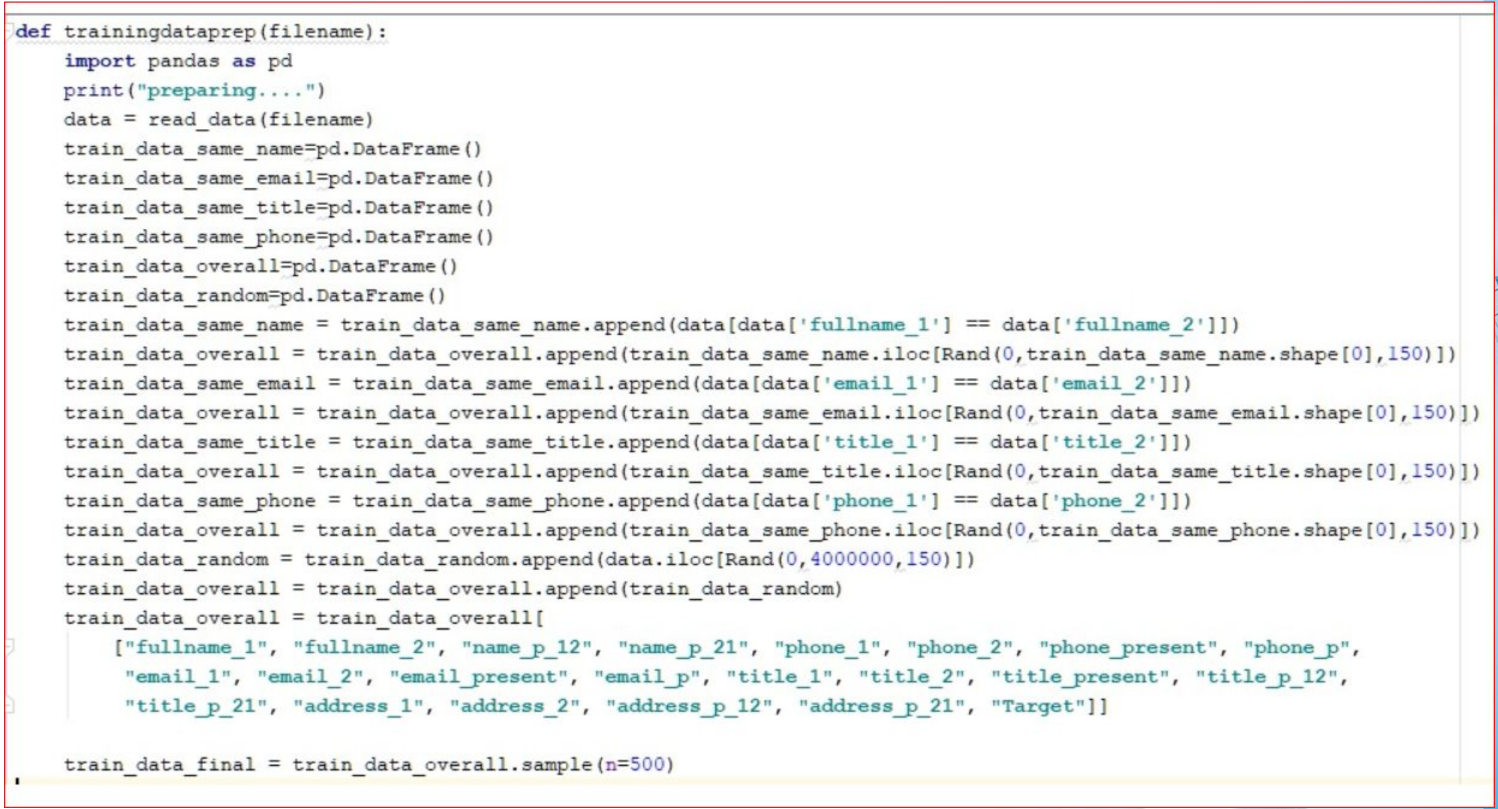
Save the Model for Test Set

DATA PRE-PROCESSING



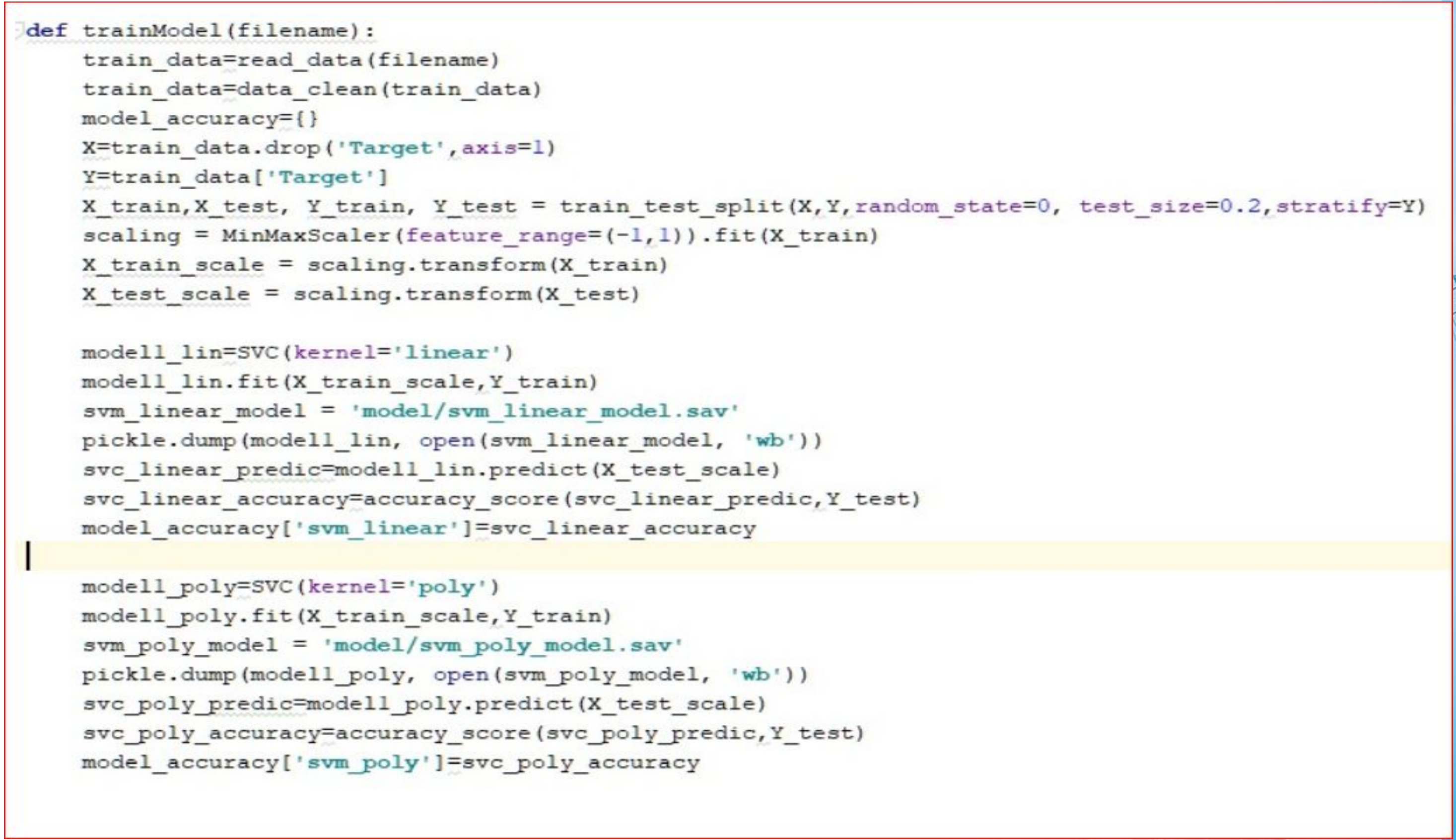
```
# Performing Name Matching
try:
    n12, n21 = entity_matching(df["FullName"][i], df["FullName"][j])
    outdict["name_p_12"] = encode_percentages(n12)
    outdict["name_p_21"] = encode_percentages(n21)
except Exception as e:
    print("Exception in finding Name Matching : ",e)
    pass

# Performing Address Matching
try:
    n12, n21 = entity_matching(address1, address2)
    outdict["address_p_12"] = encode_percentages(n12)
    outdict["address_p_21"] = encode_percentages(n21)
except Exception as e:
    print("Exception in finding Name Matching : ",e)
    pass

# Performing Email Matching
try:
    if df["Email"][i] != "None" and df["Email"][j]!="None":
        ell = string_matching(df["Email"][i], df["Email"][j])
        outdict["email_present"] = 1
        outdict["email_p"]=encode_percentages(ell)
except Exception as e:
    print("Exception in finding Email Matching : ",e)
    pass
```

TRAINING DATA PREPARATION

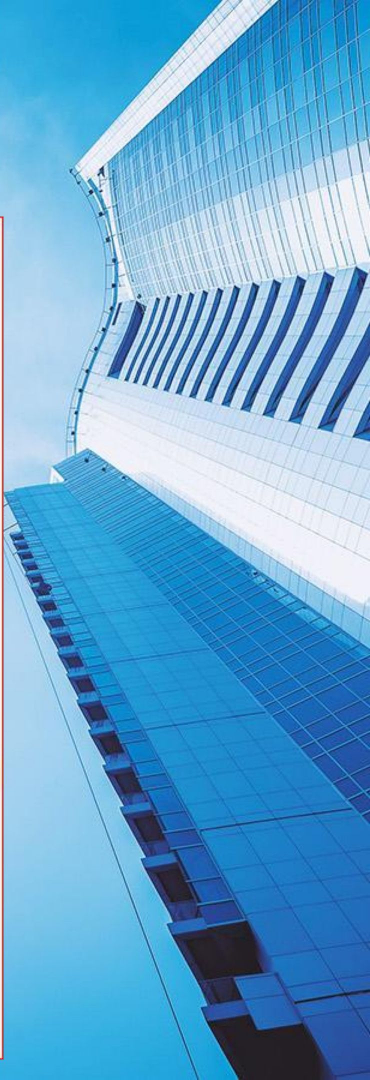


```python
def trainingdataprep(filename):
    import pandas as pd
    print("preparing....")
    data = read_data(filename)
    train_data_same_name=pd.DataFrame()
    train_data_same_email=pd.DataFrame()
    train_data_same_title=pd.DataFrame()
    train_data_same_phone=pd.DataFrame()
    train_data_overall=pd.DataFrame()
    train_data_random=pd.DataFrame()
    train_data_same_name = train_data_same_name.append(data[data['fullname_1'] == data['fullname_2']])
    train_data_overall = train_data_overall.append(train_data_same_name.iloc[Rand(0,train_data_same_name.shape[0],150)])
    train_data_same_email = train_data_same_email.append(data[data['email_1'] == data['email_2']])
    train_data_overall = train_data_overall.append(train_data_same_email.iloc[Rand(0,train_data_same_email.shape[0],150)])
    train_data_same_title = train_data_same_title.append(data[data['title_1'] == data['title_2']])
    train_data_overall = train_data_overall.append(train_data_same_title.iloc[Rand(0,train_data_same_title.shape[0],150)])
    train_data_same_phone = train_data_same_phone.append(data[data['phone_1'] == data['phone_2']])
    train_data_overall = train_data_overall.append(train_data_same_phone.iloc[Rand(0,train_data_same_phone.shape[0],150)])
    train_data_random = train_data_random.append(data.iloc[Rand(0,4000000,150)])
    train_data_overall = train_data_overall.append(train_data_random)
    train_data_overall = train_data_overall[
        ["fullname_1", "fullname_2", "name_p_12", "name_p_21", "phone_1", "phone_2", "phone_present", "phone_p",
         "email_1", "email_2", "email_present", "email_p", "title_1", "title_2", "title_present", "title_p_12",
         "title_p_21", "address_1", "address_2", "address_p_12", "address_p_21", "Target"]]

    train_data_final = train_data_overall.sample(n=500)
```

MODEL TRAINING

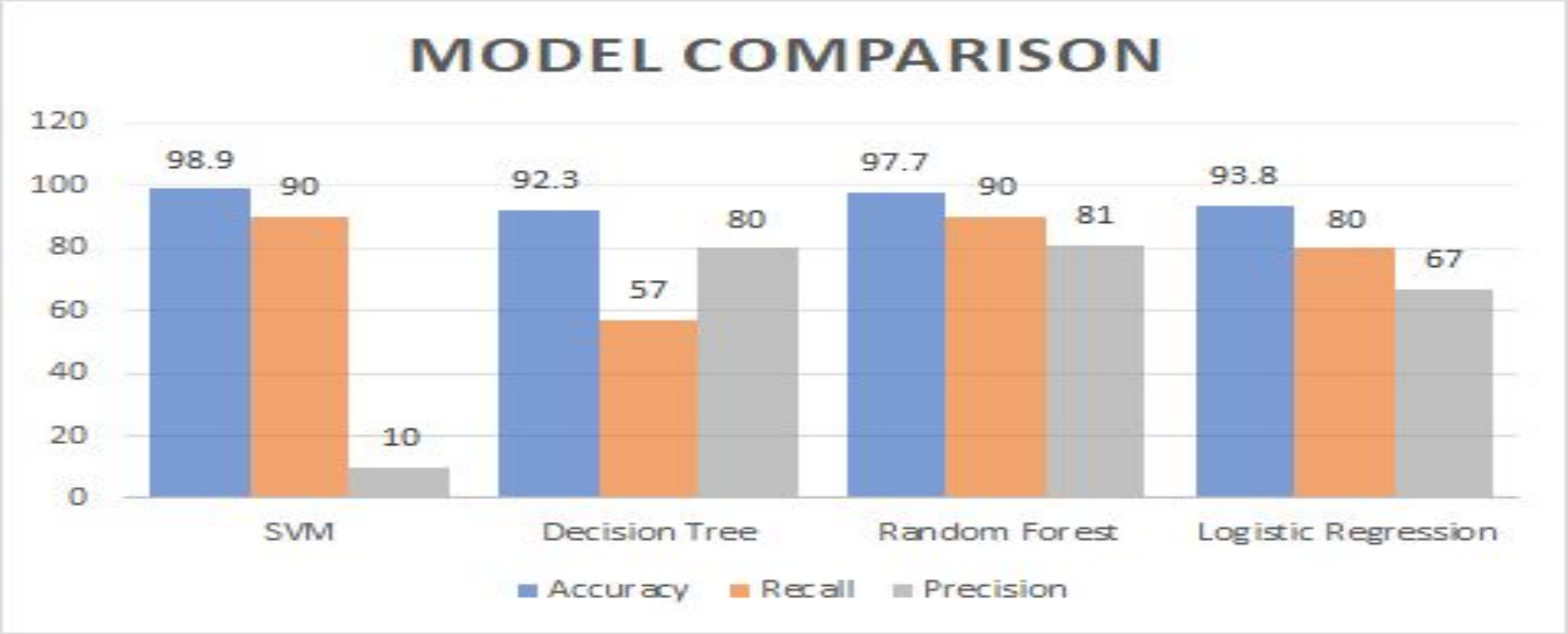


```
def trainModel(filename):
    train_data=read_data(filename)
    train_data=data_clean(train_data)
    model_accuracy={}
    X=train_data.drop('Target',axis=1)
    Y=train_data['Target']
    X_train,X_test, Y_train, Y_test = train_test_split(X,Y,random_state=0, test_size=0.2,stratify=Y)
    scaling = MinMaxScaler(feature_range=(-1,1)).fit(X_train)
    X_train_scale = scaling.transform(X_train)
    X_test_scale = scaling.transform(X_test)

    modell_lin=SVC(kernel='linear')
    modell_lin.fit(X_train_scale,Y_train)
    svm_linear_model = 'model/svm_linear_model.sav'
    pickle.dump(modell_lin, open(svm_linear_model, 'wb'))
    svc_linear_predic=modell_lin.predict(X_test_scale)
    svc_linear_accuracy=accuracy_score(svc_linear_predic,Y_test)
    model_accuracy['svm_linear']=svc_linear_accuracy

    modell_poly=SVC(kernel='poly')
    modell_poly.fit(X_train_scale,Y_train)
    svm_poly_model = 'model/svm_poly_model.sav'
    pickle.dump(modell_poly, open(svm_poly_model, 'wb'))
    svc_poly_predic=modell_poly.predict(X_test_scale)
    svc_poly_accuracy=accuracy_score(svc_poly_predic,Y_test)
    model_accuracy['svm_poly']=svc_poly_accuracy
```

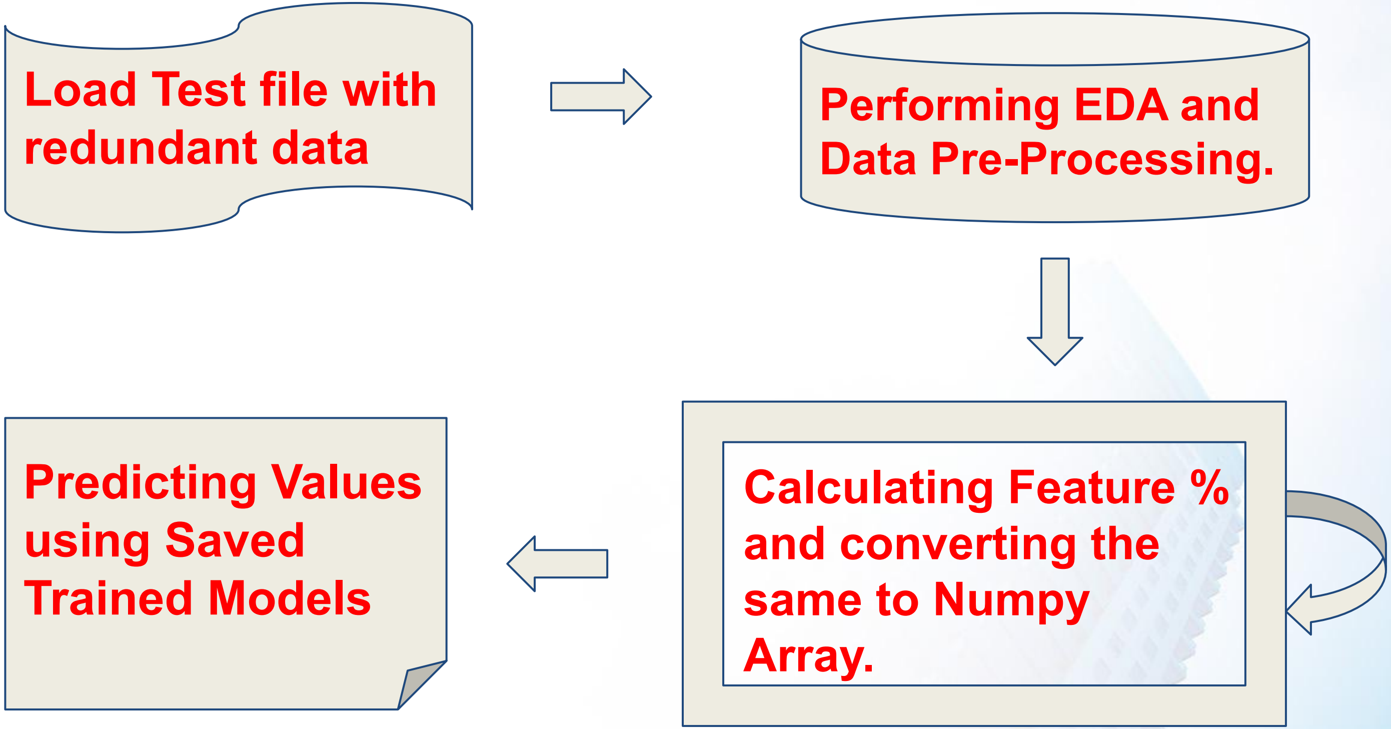


MODEL SELECTION:

Classification Algorithms Used



PHASE 3: TESTING AND PREDICTION



Load Test file with redundant data

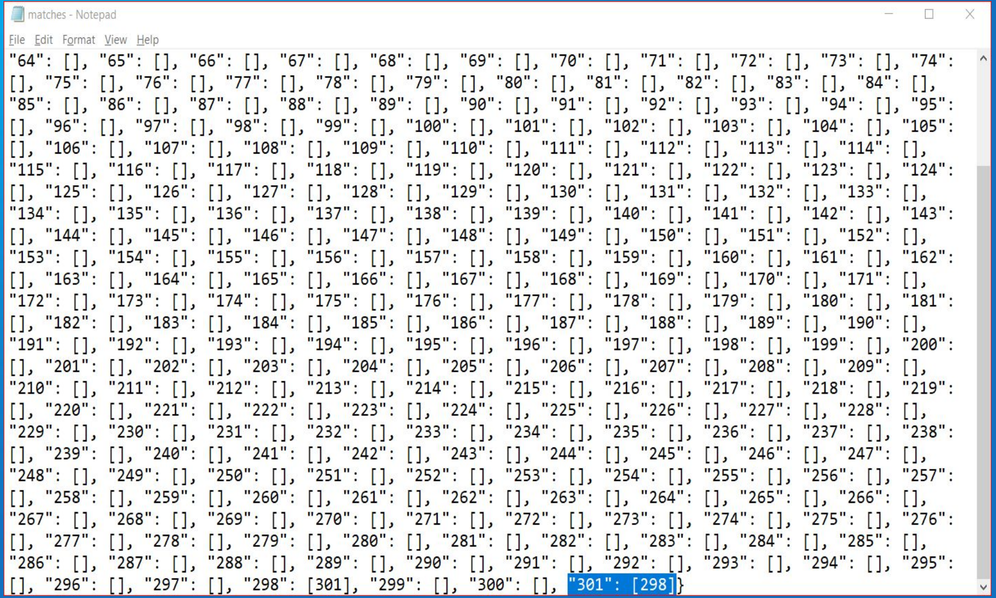
Performing EDA and Data Pre-Processing.

Calculating Feature % and converting the same to Numpy Array.

Predicting Values using Saved Trained Models

REDUNDANT DATA SEGREGATION

The output of the Machine Learning Algorithm is stored in the form of a **DICTIONARY FORMAT** with a **key-value pair.**



```
"64": [], "65": [], "66": [], "67": [], "68": [], "69": [], "70": [], "71": [], "72": [], "73": [], "74":
[], "75": [], "76": [], "77": [], "78": [], "79": [], "80": [], "81": [], "82": [], "83": [], "84": [],
"85": [], "86": [], "87": [], "88": [], "89": [], "90": [], "91": [], "92": [], "93": [], "94": [], "95":
[], "96": [], "97": [], "98": [], "99": [], "100": [], "101": [], "102": [], "103": [], "104": [], "105":
[], "106": [], "107": [], "108": [], "109": [], "110": [], "111": [], "112": [], "113": [], "114": [],
"115": [], "116": [], "117": [], "118": [], "119": [], "120": [], "121": [], "122": [], "123": [], "124":
[], "125": [], "126": [], "127": [], "128": [], "129": [], "130": [], "131": [], "132": [], "133": [],
"134": [], "135": [], "136": [], "137": [], "138": [], "139": [], "140": [], "141": [], "142": [], "143":
[], "144": [], "145": [], "146": [], "147": [], "148": [], "149": [], "150": [], "151": [], "152": [],
"153": [], "154": [], "155": [], "156": [], "157": [], "158": [], "159": [], "160": [], "161": [], "162":
[], "163": [], "164": [], "165": [], "166": [], "167": [], "168": [], "169": [], "170": [], "171": [],
"172": [], "173": [], "174": [], "175": [], "176": [], "177": [], "178": [], "179": [], "180": [], "181":
[], "182": [], "183": [], "184": [], "185": [], "186": [], "187": [], "188": [], "189": [], "190": [],
"191": [], "192": [], "193": [], "194": [], "195": [], "196": [], "197": [], "198": [], "199": [], "200":
[], "201": [], "202": [], "203": [], "204": [], "205": [], "206": [], "207": [], "208": [], "209": [],
"210": [], "211": [], "212": [], "213": [], "214": [], "215": [], "216": [], "217": [], "218": [], "219":
[], "220": [], "221": [], "222": [], "223": [], "224": [], "225": [], "226": [], "227": [], "228": [],
"229": [], "230": [], "231": [], "232": [], "233": [], "234": [], "235": [], "236": [], "237": [], "238":
[], "239": [], "240": [], "241": [], "242": [], "243": [], "244": [], "245": [], "246": [], "247": [],
"248": [], "249": [], "250": [], "251": [], "252": [], "253": [], "254": [], "255": [], "256": [], "257":
[], "258": [], "259": [], "260": [], "261": [], "262": [], "263": [], "264": [], "265": [], "266": [],
"267": [], "268": [], "269": [], "270": [], "271": [], "272": [], "273": [], "274": [], "275": [], "276":
[], "277": [], "278": [], "279": [], "280": [], "281": [], "282": [], "283": [], "284": [], "285": [],
"286": [], "287": [], "288": [], "289": [], "290": [], "291": [], "292": [], "293": [], "294": [], "295":
[], "296": [], "297": [], "298": [301], "299": [], "300": [], "301": [298]}
```

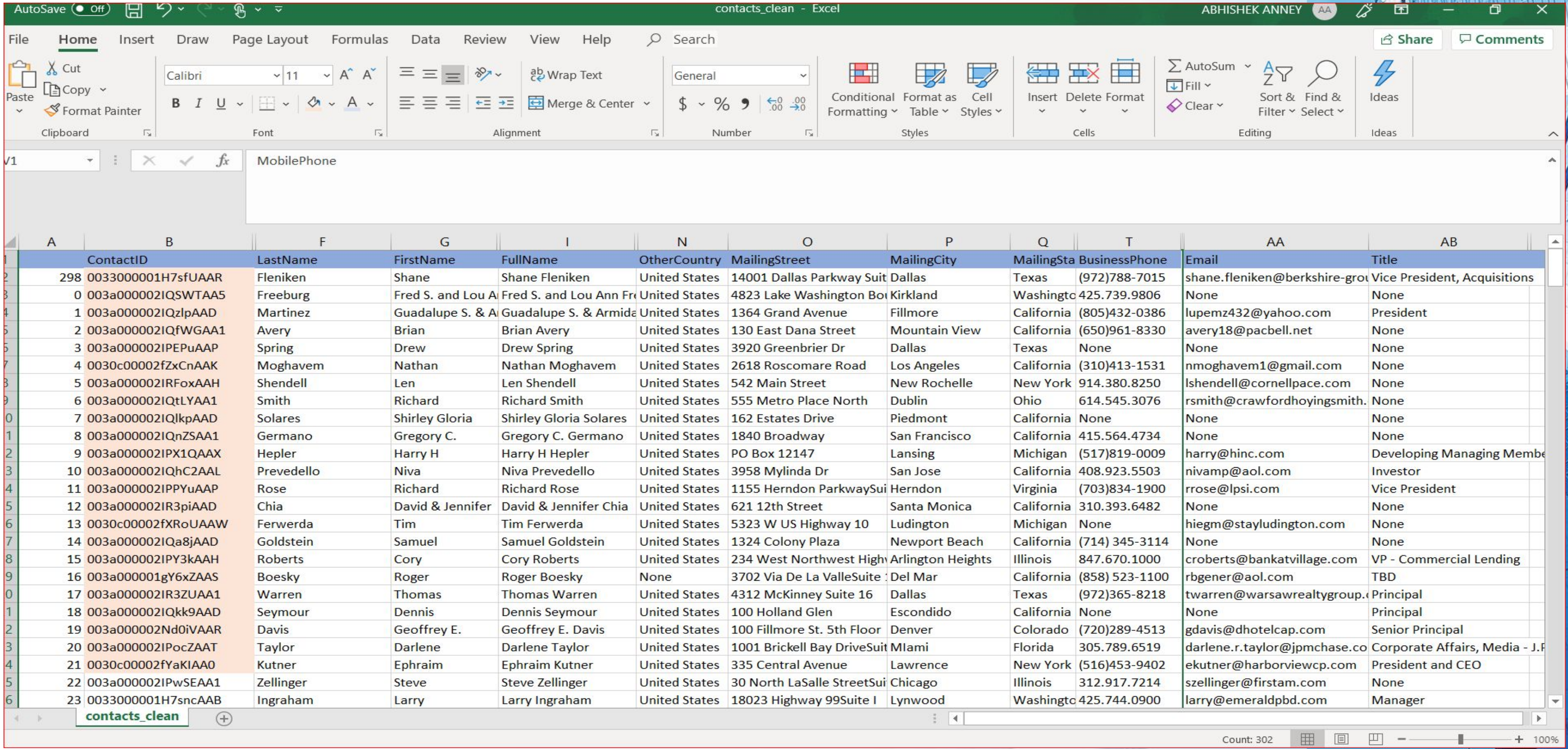
METRICS OF DUPLICATION:

Basket analysis has been done to optimize the process of Data Acquisition and Data-Entry.

3 segments were created for the data:

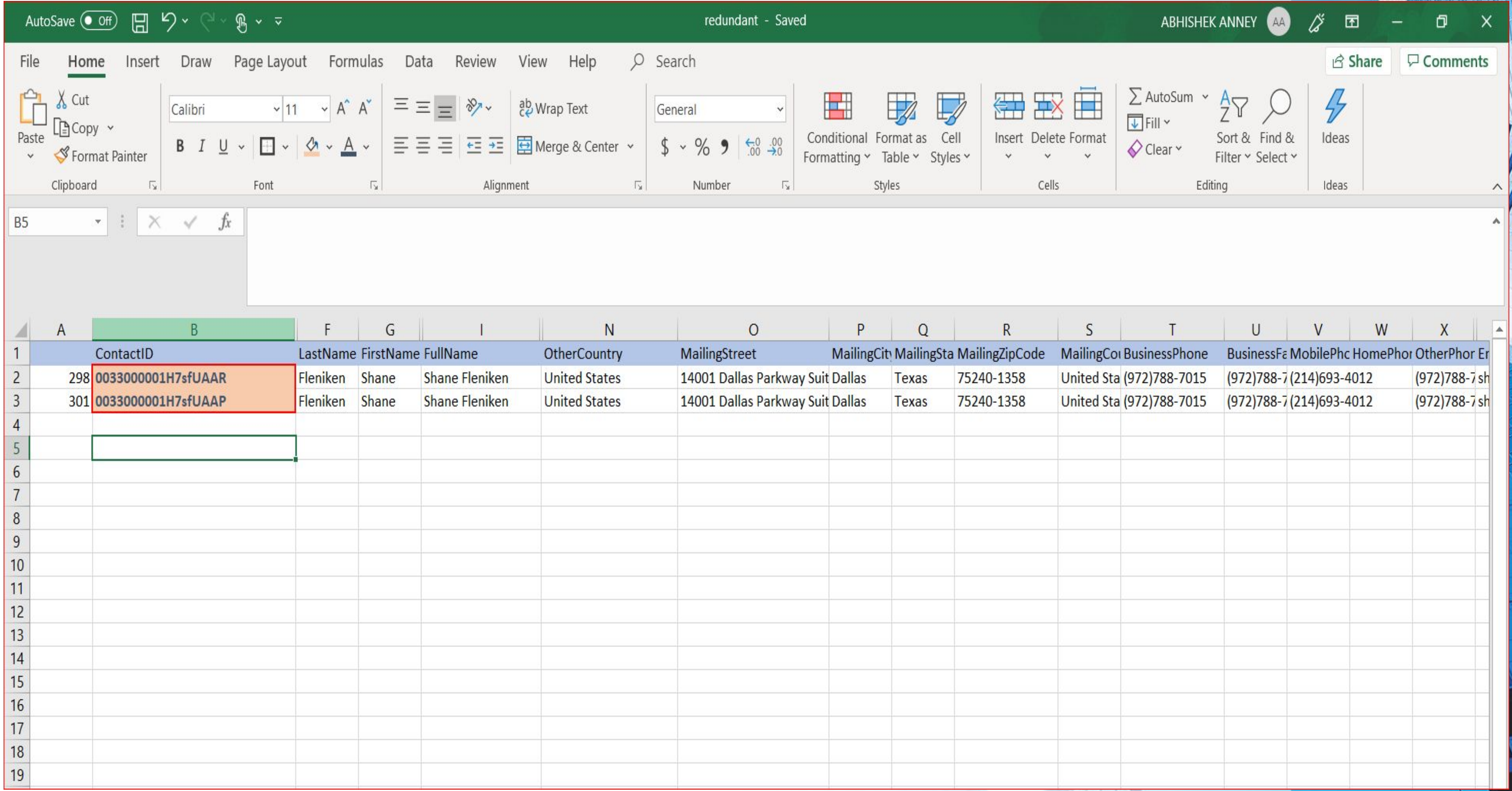
- 0%-50% Match: Basket 1
- 50%-85% Match: Basket 2
- 85%-100% Match: Basket 3

CLEAN DATA OUTPUT :



A	B	F	G	I	N	O	P	Q	T	AA	AB
	ContactID	LastName	FirstName	FullName	OtherCountry	MailingStreet	MailingCity	MailingSta	BusinessPhone	Email	Title
298	0033000001H7sfUAAR	Fleniken	Shane	Shane Fleniken	United States	14001 Dallas Parkway Suit	Dallas	Texas	(972)788-7015	shane.fleniken@berkshire-grou	Vice President, Acquisitions
0	003a000002IQSWTAA5	Freeburg	Fred S. and Lou A	Fred S. and Lou Ann Fr	United States	4823 Lake Washington Bou	Kirkland	Washingtc	425.739.9806	None	None
1	003a000002IQzlpAAD	Martinez	Guadalupe S. & A	Guadalupe S. & Armida	United States	1364 Grand Avenue	Fillmore	California	(805)432-0386	lupemz432@yahoo.com	President
2	003a000002IQfWGAA1	Avery	Brian	Brian Avery	United States	130 East Dana Street	Mountain View	California	(650)961-8330	avery18@pacbell.net	None
3	003a000002IPEPuAAP	Spring	Drew	Drew Spring	United States	3920 Greenbrier Dr	Dallas	Texas	None	None	None
4	0030c00002fZxCnAAK	Moghavem	Nathan	Nathan Moghavem	United States	2618 Roscomare Road	Los Angeles	California	(310)413-1531	nmoghavem1@gmail.com	None
5	003a000002IRFoxAAH	Shendell	Len	Len Shendell	United States	542 Main Street	New Rochelle	New York	914.380.8250	lshendell@cornellpace.com	None
6	003a000002IQtLYAA1	Smith	Richard	Richard Smith	United States	555 Metro Place North	Dublin	Ohio	614.545.3076	rsmith@crawfordhoyingsmith.	None
7	003a000002IQlkpAAD	Solares	Shirley Gloria	Shirley Gloria Solares	United States	162 Estates Drive	Piedmont	California	None	None	None
8	003a000002IQnZSAA1	Germano	Gregory C.	Gregory C. Germano	United States	1840 Broadway	San Francisco	California	415.564.4734	None	None
9	003a000002IPX1QAAX	Hepler	Harry H	Harry H Hepler	United States	PO Box 12147	Lansing	Michigan	(517)819-0009	harry@hinc.com	Developing Managing Membe
10	003a000002IQhC2AAL	Prevedello	Niva	Niva Prevedello	United States	3958 Mylinda Dr	San Jose	California	408.923.5503	nivamp@aol.com	Investor
11	003a000002IPPYuAAP	Rose	Richard	Richard Rose	United States	1155 Herndon ParkwaySui	Herndon	Virginia	(703)834-1900	rrose@lpsi.com	Vice President
12	003a000002IR3piAAD	Chia	David & Jennifer	David & Jennifer Chia	United States	621 12th Street	Santa Monica	California	310.393.6482	None	None
13	0030c00002fXRoUAAW	Ferwerda	Tim	Tim Ferwerda	United States	5323 W US Highway 10	Ludington	Michigan	None	hiegm@stayludington.com	None
14	003a000002IQa8jAAD	Goldstein	Samuel	Samuel Goldstein	United States	1324 Colony Plaza	Newport Beach	California	(714) 345-3114	None	None
15	003a000002IPY3kAAH	Roberts	Cory	Cory Roberts	United States	234 West Northwest High	Arlington Heights	Illinois	847.670.1000	croberts@bankatvillage.com	VP - Commercial Lending
16	003a000001gY6xZAAS	Boesky	Roger	Roger Boesky	None	3702 Via De La ValleSuite	Del Mar	California	(858) 523-1100	rbgener@aol.com	TBD
17	003a000002IR3ZUAA1	Warren	Thomas	Thomas Warren	United States	4312 McKinney Suite 16	Dallas	Texas	(972)365-8218	twarren@warsawrealtygroup.	Principal
18	003a000002IQkk9AAD	Seymour	Dennis	Dennis Seymour	United States	100 Holland Glen	Escondido	California	None	None	Principal
19	003a000002Nd0iVAAR	Davis	Geoffrey E.	Geoffrey E. Davis	United States	100 Fillmore St. 5th Floor	Denver	Colorado	(720)289-4513	gdavis@dhotelcap.com	Senior Principal
20	003a000002IPocZAAT	Taylor	Darlene	Darlene Taylor	United States	1001 Brickell Bay DriveSuit	MIami	Florida	305.789.6519	darlene.r.taylor@jpmchase.co	Corporate Affairs, Media - J.F
21	0030c00002fYaKIAA0	Kutner	Ephraim	Ephraim Kutner	United States	335 Central Avenue	Lawrence	New York	(516)453-9402	ekutner@harborviewcp.com	President and CEO
22	003a000002IPwSEAA1	Zellinger	Steve	Steve Zellinger	United States	30 North LaSalle StreetSui	Chicago	Illinois	312.917.7214	szellinger@firstam.com	None
23	0033000001H7sncAAB	Ingraham	Larry	Larry Ingraham	United States	18023 Highway 99Suite I	Lynwood	Washingtc	425.744.0900	larry@emeraldpbd.com	Manager

REDUNDANT DATA OUTPUT



	A	B	F	G	I	N	O	P	Q	R	S	T	U	V	W	X	
1		ContactID	LastName	FirstName	FullName	OtherCountry	MailingStreet	MailingCit	MailingSta	MailingZipCode	MailingCo	BusinessPhone	BusinessFa	MobilePhc	HomePhoı	OtherPhor	Er
2	298	0033000001H7sfUAAR	Fleniken	Shane	Shane Fleniken	United States	14001 Dallas Parkway Suit	Dallas	Texas	75240-1358	United Sta	(972)788-7015	(972)788-7	(214)693-4012		(972)788-7	sh
3	301	0033000001H7sfUAAP	Fleniken	Shane	Shane Fleniken	United States	14001 Dallas Parkway Suit	Dallas	Texas	75240-1358	United Sta	(972)788-7015	(972)788-7	(214)693-4012		(972)788-7	sh

FINAL DELIVERY PHASE:

- Processed 3K [Sampled] rows
- Yet to process 197K rows
- Project Output includes 2 Datasets:
    1. Redundant Data
    2. Clean Data

BARRIERS AND CHALLENGES

Large Dataset - Multiple Batch Processing - Higher Run-time

KEY REQUIREMENT:

Minimum configuration for testing- **I5 with 12GB RAM and 2.7 Ghz Processor. [GPU Will be preferable]**

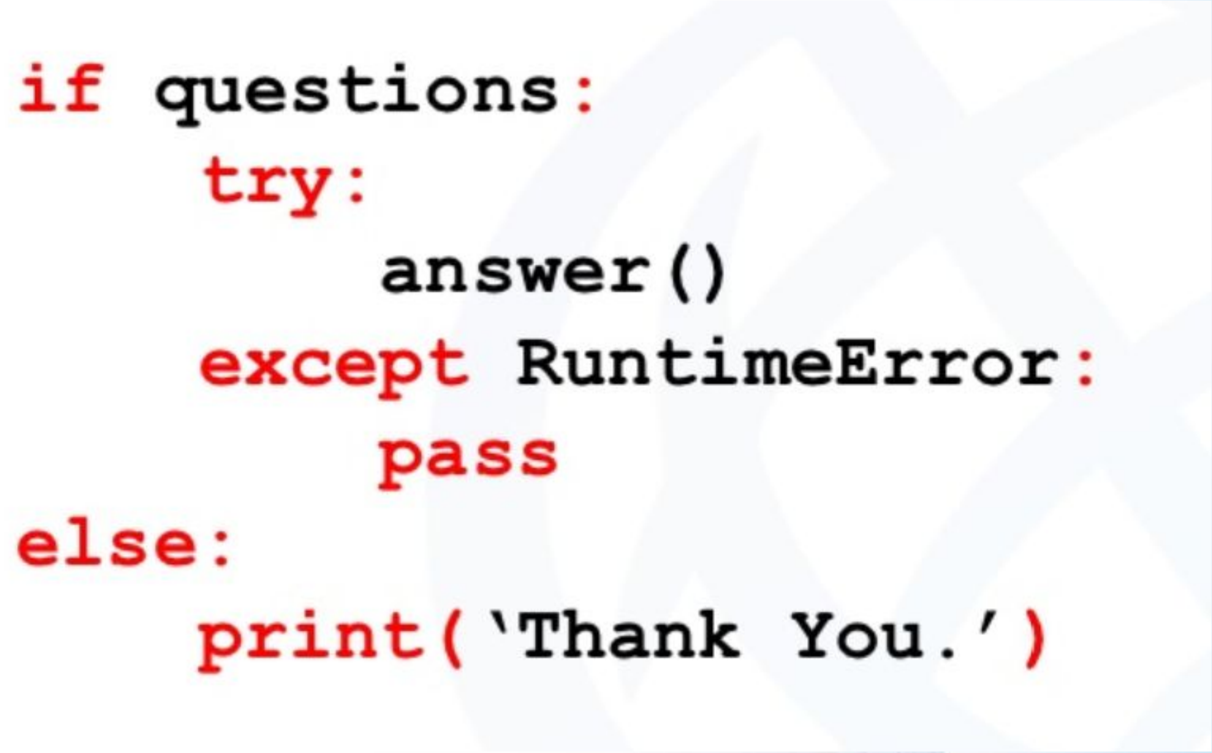
IDEA FOR FUTURE ANALYSIS

- Address verification using Google Map API Integration

- Email Verification using Zero Bounce API

O
P
[
Q
&
A
>
?

?



```
if questions:
    try:
        answer()
    except RuntimeError:
        pass
else:
    print(‘Thank You.’)
```