Email Classification Project Proposal

LIANG, Ting-Yu

Table of contents

01

Current Situation

Current Situation

Spam

1. Promotional
2. Social Media
3. Hacked

Priority

Tons of emails hit our mailbox daily, resulting in important emails being missed.

02

Solution

Objective

Classify Emails Into Different Categories

Technologies

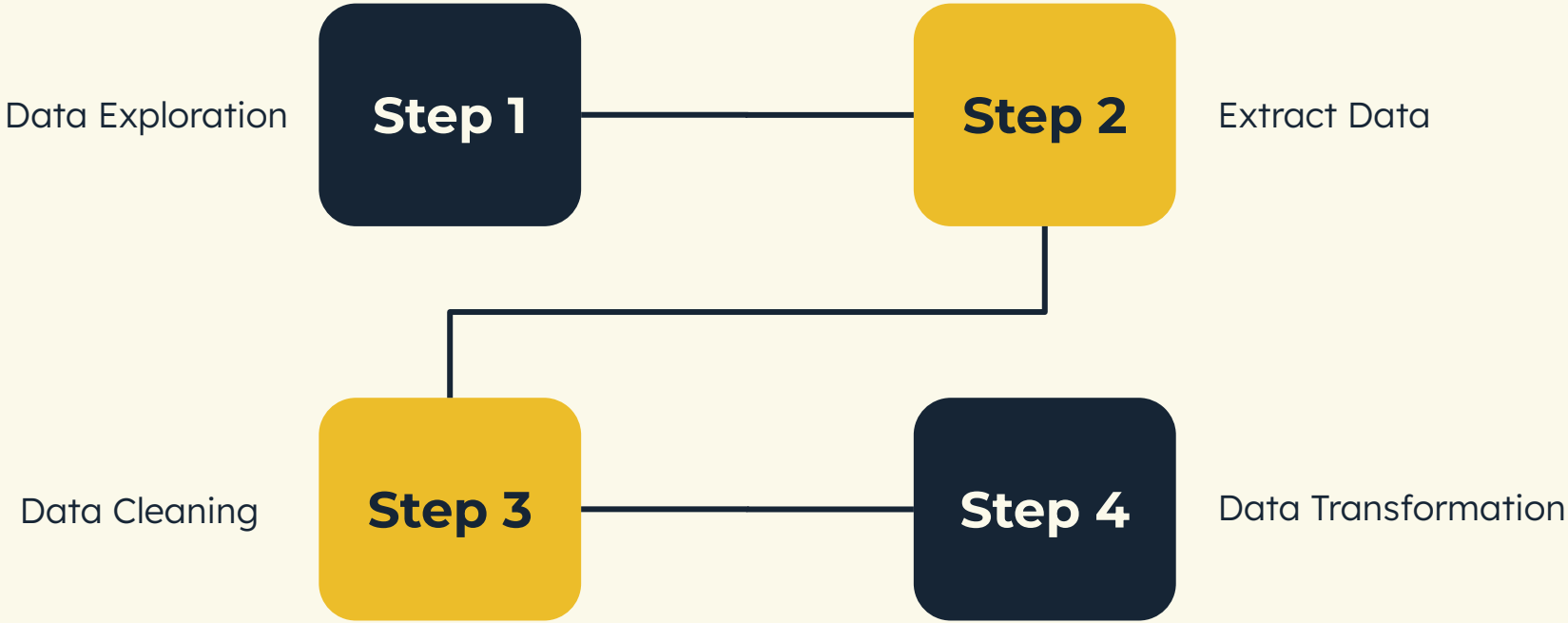
Scikit-Learn

1. Preprocessing
2. Classification
3. Regression

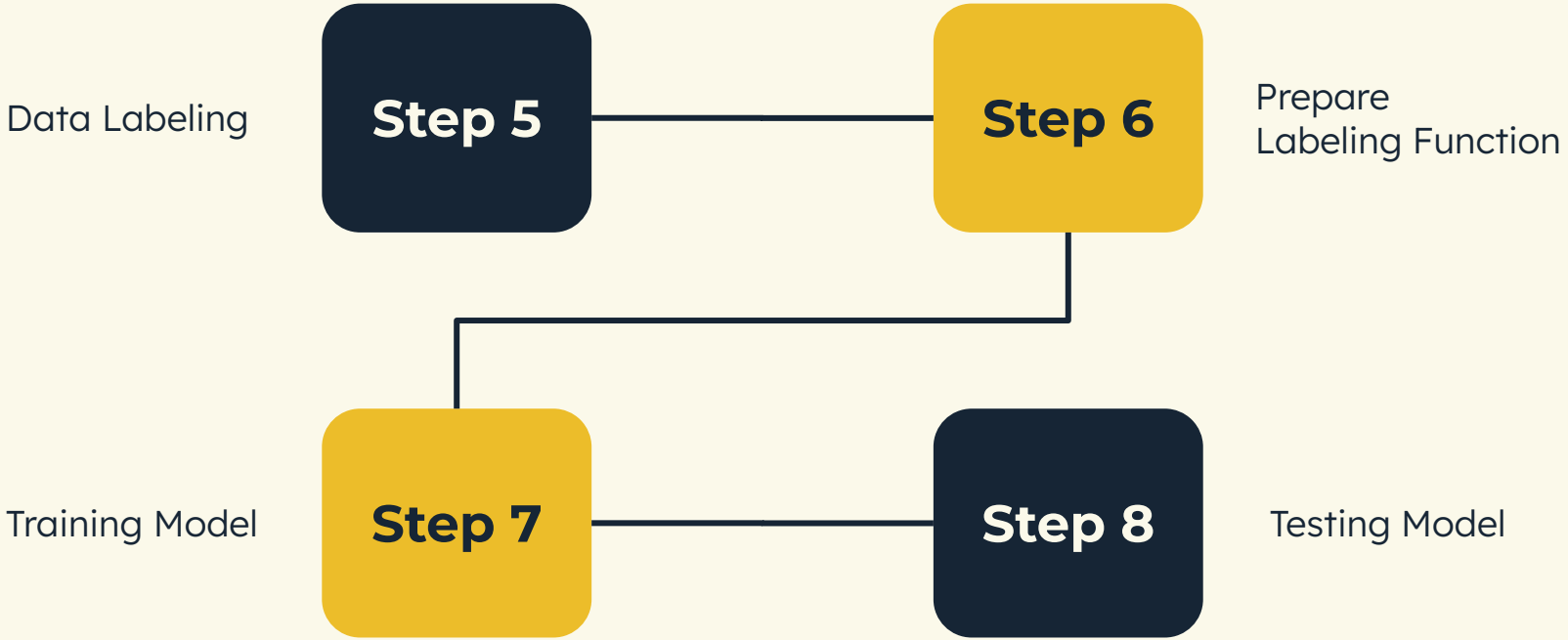
Snorkel

1. Data Labelling
2. Build training dataset

Process


Data Exploration
Step 1
Step 2
Extract Data
Data Cleaning
Step 3
Step 4
Data Transformation

Process


Data Labeling
Step 5
Step 6
Prepare
Labeling Function
Training Model
Step 7
Step 8
Testing Model

03

Demo

Step 1 - Data Exploration

```
import numpy as np
import pandas as pd
import email
```

Step 1 - Data Exploration

```
emails = pd.read_csv("emails.csv")
emails.head()
```

	file	message
0	allen-p/_sent_mail/1.	Message-ID: <18782981.1075855378110.JavaMail.e...
1	allen-p/_sent_mail/10.	Message-ID: <15464986.1075855378456.JavaMail.e...
2	allen-p/_sent_mail/100.	Message-ID: <24216240.1075855687451.JavaMail.e...
3	allen-p/_sent_mail/1000.	Message-ID: <13505866.1075863688222.JavaMail.e...
4	allen-p/_sent_mail/1001.	Message-ID: <30922949.1075863688243.JavaMail.e...

Step 1 - Data Exploration

```
print(emails.loc[0]["message"])
```

```
Message-ID: <18782981.1075855378110.JavaMail.evans@thyme>
Date: Mon, 14 May 2001 16:39:00 -0700 (PDT)
From: phillip.allen@enron.com
To: tim.belden@enron.com
Subject:
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-From: Phillip K Allen
X-To: Tim Belden <Tim Belden/Enron@EnronXGate>
X-cc:
X-bcc:
X-Folder: \Phillip_Allen_Jan2002_1\Allen, Phillip K.\'Sent Mail
X-Origin: Allen-P
X-FileName: pallen (Non-Privileged).pst

Here is our forecast
```

Step 2 - Extract Data

```
#Extract Headers
def get_field(field, messages):
    column = []
    for message in messages:
        e = email.message_from_string(message)
        column.append(e.get(field))
    return column
```

Step 2 - Extract Data

```
emails['date'] = get_field("Date", emails['message'])
emails['subject'] = get_field("Subject", emails['message'])
emails['XFolder'] = get_field("X-Folder", emails['message'])
emails['X-From'] = get_field("X-From", emails['message'])
emails['X-To'] = get_field("X-To", emails['message'])
emails.head(3)
```

	file	message	date	subject	XFolder	X-From	X-To
0	allen-p/_sent_mail/1.	Message-ID: <18782981.1075855378110.JavaMail.e...	Mon, 14 May 2001 16:39:00 -0700 (PDT)		\Phillip_Allen_Jan2002_1\Allen, Phillip K.\'Se...	Phillip K Allen	Tim Belden <Tim Belden/Enron@EnronXGate>
1	allen-p/_sent_mail/10.	Message-ID: <15464986.1075855378456.JavaMail.e...	Fri, 4 May 2001 13:51:00 -0700 (PDT)	Re:	\Phillip_Allen_Jan2002_1\Allen, Phillip K.\'Se...	Phillip K Allen	John J Lavorato <John J Lavorato/ENRON@enronXg...
2	allen-p/_sent_mail/100.	Message-ID: <24216240.1075855687451.JavaMail.e...	Wed, 18 Oct 2000 03:00:00 -0700 (PDT)	Re: test	\Phillip_Allen_Dec2000\Notes Folders\'sent mail	Phillip K Allen	Leah Van Arsdall

Step 2 - Extract Data

```
#Extract Message Body
def body(messages):
    column = []
    for message in messages:
        e = email.message_from_string(message)
        column.append(e.get_payload())
    return column
emails['body'] = body(emails['message'])
emails['body'].head(3)
```

```
0                                 Here is our forecast\n\n
1    Traveling to have a business meeting takes the...
2                          test successful.  way to go!!!
Name: body, dtype: object
```

Step 2 - Extract Data

```
#Extract Employee Name
def employee(file):
    column = []
    for string in file:
        column.append(string.split("/")[0])
    return column

emails['employee'] = employee(emails['file'])
emails['employee'].head(3)
```

```
0    allen-p
1    allen-p
2    allen-p
Name: employee, dtype: object
```

Step 3 - Data Cleaning

```
#Convert date column to datetime
import datetime
from dateutil import parser
x = parser.parse("Fri, 4 May 2001 13:51:00 -0700
(PDT)")
print(x.strftime("%d-%m-%Y %H:%M:%S"))
```

```
04-05-2001 13:51:00
```

Step 3 - Data Cleaning

```
def change_type(dates):
    column = []
    for date in dates:
        column.append(parser.parse(date).strftime("%d-%m-%Y %H:%M:%S"))
    return column
emails['date'] = change_type(emails['date'])
```

Step 3 - Data Cleaning

```
print(emails['XFolder'][0])
emails['XFolder'][0].split("\\")[-1]
```

```
\Phillip_Allen_Jan2002_1\Allen, Phillip K.\'Sent Mail
"'Sent Mail"
```

Step 3 - Data Cleaning

```
#Extract the last folder name
def preprocess_folder(folders):
    column = []
    for folder in folders:
        if (folder is None or folder == ""):
            column.append(np.nan)
        else:
            column.append(folder.split("\\")[-1].lower())
    return column
emails['XFolder'] = preprocess_folder(emails['XFolder'])
```

```
103829    bankruptcy
103830    bankruptcy
103831    bankruptcy
Name: XFolder, dtype: object
```

Step 3 - Data Cleaning

```
#Remove folders containing too few emails (less than 2)
unwanted_folders = ["all documents", "deleted items", "discussion threads",
"sent", "deleted Items", "inbox","sent items", "'sent mail", "untitled", "notes
inbox", "junk file", "calendar"]
emails = emails.loc[~emails['XFolder'].isin(unwanted_folders)]
```

Step 3 - Data Cleaning

```
#Replace empty missing values in subject with np.nan
def replace_empty_with_nan(subject):
    column = []
    for val in subject:
        if (val == ""):
            column.append(np.nan)
        else:
            column.append(val)
    return column
```

```
emails['subject'] = replace_empty_with_nan(emails['subject'])
emails['X-To'] = replace_empty_with_nan(emails['X-To'])
```

Step 3 - Data Cleaning

```
#Drop missing value rows
emails.dropna(axis=0, inplace=True)
```

```
emails.isnull().sum()
```

```
file          0
message       0
date          0
subject       0
XFolder       0
X-From        0
X-To          0
body          0
employee      0
dtype: int64
```

Step 3 - Data Cleaning

```
emails.head()
```

	file	message	date	subject	XFolder	X-From	X-To	body	employee
3026	allen-p/straw/1.	Message-ID: <12644875.1075855692817.JavaMail.e...	14-03-2000 17:07:00	Central Texas Bale Resource	straw	bobregon@bga.com	list <strawbale@crest.org>	Hi All\n\nWe are looking for a wheat farmer ne...	allen-p
3027	allen-p/straw/2.	Message-ID: <22208447.1075855692838.JavaMail.e...	17-02-2000 07:37:00	Re: History of Lime and Cement	straw	rob_tom@freenet.carleton.ca (Robert W. Tom)	CALXA@aol.com	'Arry (calxa@aol.com), Lime Ex-splurt Extraord...	allen-p
3028	allen-p/straw/3.	Message-ID: <31438311.1075855692860.JavaMail.e...	17-02-2000 07:01:00	History of Lime and Cement	straw	CALXA@aol.com	strawbale@crest.org, absteen@dakotacom.net	Folks,\n\nI just found this interesting site a...	allen-p
3029	allen-p/straw/4.	Message-ID: <2055670.1075855692881.JavaMail.ev...	10-02-2000 01:46:00	Re: Newsgroups	straw	billc@greenbuilder.com	strawbale@crest.org	>What other cool newsgroups are available for ...	allen-p
3030	allen-p/straw/5.	Message-ID: <22141218.1075855692903.JavaMail.e...	07-01-2000 16:29:00	RE: concrete stain	straw	"Matt" <matt@fastpacket.net>	strawbale@crest.org	> Hi,\n> We recently faced the same questions ...	allen-p

Step 3 - Data Cleaning

```
cols_to_drop = ['file','message','date','X-From','X-To']
```

```
emails.drop(cols_to_drop, axis=1, inplace=True)
```

```
emails.head()
```

	subject	XFolder	body	employee
103829	RE: contracts and credit	bankruptcy	Thanks--I'll include it in the master file. \...	farmer-d
103830	Northern Natural Gas	bankruptcy	Yesterday, Enron settled a procedural dispute ...	farmer-d
103831	FW: Customers	bankruptcy	\n\n -----Original Message-----\nFrom: \tBuss,...	farmer-d
103832	Contract status needed	bankruptcy	On Monday we have to file a "transition plan" ...	farmer-d
103833	FW:	bankruptcy	Listed below are some guidelines for gathering...	farmer-d

Step 3 - Data Cleaning

```
#Extract emails for employees who had over 2000 emails
email_count = dict(emails["employee"].value_counts())
reduced_emails = [key for key, val in email_count.items() if val >= 2000]
emails = emails.loc[emails['employee'].isin(reduced_emails)]
print(emails["employee"].value_counts())
```

```
kean-s          5344
kaminski-v      4388
kitchen-l       3775
farmer-d        3727
williams-w3     2740
lokay-m         2422
taylor-m        2374
Name: employee, dtype: int64
```

Step 3 - Data Cleaning

```
#Choose an employee
employee = emails[emails["employee"] == "farmer-d"]
```

```
#Returns the folders containing more than 'n' number of emails
def remove_folders(emails, n):
    email_count = dict(emails["XFolder"].value_counts())
    small_folders = [key for key, val in email_count.items() if val <= n]
    emails = emails.loc[~emails['XFolder'].isin(small_folders)]
    return emails
```

```
n = 200
employee = remove_folders(employee, n)
```

Step 4 - Data Transformation

TfidVectorizer:

Transforms text to feature vectors that can be used as input to estimator.

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(sublinear_tf=True, min_df=5, norm='l2',encoding='latin-1',
ngram_range=(1, 2), stop_words='english')
features = tfidf.fit_transform(employee.body).toarray()
labels = employee.label
```

Step 5 - Data Labeling

LabelEncoder: Converting each value in a column to a number

```
#Encoding class labels
from sklearn.preprocessing import LabelEncoder
def label_encoder(df):
    class_le = LabelEncoder()
    # apply label encoder on the `XFolder` column
    y = class_le.fit_transform(df['XFolder'])
    df.loc[:,'label'] = y
    return df
```

```
label_encoder(employee)
unique_folders = employee["label"].unique()
print (unique_folders)
```

```
[2 0 1 3 4]
```

Step 5 - Data Labeling

```
employee.head()
```

	subject	XFolder	body	employee	label
104628	Registration Welcome Email	personal	\n Thank you for registering at Citibank Acco...	farmer-d	2
104629	NEON Lesson 5	personal	Please respond to Here is lesson #5. Have fun...	farmer-d	2
104630	PLEASE READ - IMPORTANT INFORMATION FOR PARTIC...	personal	Due to current business circumstances, on Nove...	farmer-d	2
104631	eConnect VPN	personal	You have been approved and added to the eConne...	farmer-d	2
104632	NEON Retreat	personal	\nHo ho ho, we're around to that most wonderf...	farmer-d	2

Step 6 - Prepare Labeling Function

```
%pip install snorkel
```

```
from snorkel.labeling import labeling_function
```

```
PERSONAL=2
INDUSTRIALS=0
LOGISTICS=1
TUFCO=3
WELLHEAD=4
ABSTAIN=-1
```

Step 6 - Prepare Labeling Function

```
#Find the terms that are the most correlated with each of the category
from sklearn.feature_selection import chi2
import numpy as np
def keyword(features,name):
    features_chi2 = chi2(features, labels == name)
    indices = np.argsort(features_chi2[0])
    feature_names = np.array(tfidf.get_feature_names())[indices]
    unigrams = [v for v in feature_names if len(v.split(' ')) == 1]
    bigrams = [v for v in feature_names if len(v.split(' ')) == 2]
    unigrams=unigrams[-25:]
    bigrams=bigrams[-25:]
    return unigrams+bigrams
```

Step 6 - Prepare Labeling Function

```
keyword(features,0)
```

```
['equiva',
 'mobil',
 'xls',
 'beaumont',
 '1373',
 '77002',
 'lloyd',
 '2700',
 '8722',
 '8659',
 'louisiana',
 'archer',
 'methanol',
 'added',
 'carbide',
 '5m',
 'ricky',
 'papayoti',
 'calpine',
 'equistar',
 '830',
 'swing',
 'buyback',
 'shell',
 'lee',
 'swing ticket',
 'suite 2700',
 '2700 houston',
 '700 louisiana',
 'supply 700',
 '77002 713',
 '830 8659',
 '830 8722',
 '8659 direct',
 'archer fuel',
 'direct 713',
 '8722 fax',
 'robert lloyd',
 'union carbide',
 '2000 activity',
 'fax calpine',
 'ricky archer',
 'thanks lee',
 'papayoti hou',
 'daily gas',
 'nomination doc',
 'gas nomination',
 'lee papayoti',
 'calpine daily',
 '713 830']
```

Step 6 - Prepare Labeling Function

```
@labeling_function()
def lf_keyword_personal(x):
 keywords=keyword(features,2)
 return PERSONAL if any(word in x.body.lower() for word in keywords) else ABSTAIN
```

```
@labeling_function()
def lf_keyword_industrials(x):
 keywords=keyword(features,0)
 return INDUSTRIALS if any(word in x.body.lower() for word in keywords) else ABSTAIN
```

```
@labeling_function()
def lf_keyword_logistics(x):
 keywords=keyword(features,1)
 return LOGISTICS if any(word in x.body.lower() for word in keywords) else ABSTAIN
```

Step 6 - Prepare Labeling Function

```
@labeling_function()
def lf_keyword_tufco(x):
 keywords=keyword(features,3)
 return TUFCO if any(word in x.body.lower() for word in keywords) else ABSTAIN
```

```
@labeling_function()
def lf_keyword_wellhead(x):
 keywords=keyword(features,4)
 return WELLHEAD if any(word in x.body.lower() for word in keywords) else ABSTAIN
```

Step 7 - Training Model

- **PandasLFApplier:** LF applier for a Pandas DataFrame.
- **MajorityLabelVoter:** Majority vote label model.

```
from sklearn.model_selection import train_test_split
train, test = train_test_split(new_data,test_size=0.2)
Y_test = test.label.values
```

```
from snorkel.labeling import PandasLFApplier
from snorkel.labeling.model import LabelModel
from snorkel.labeling.model import MajorityLabelVoter
```

Step 7 - Training Model

```
# Define the set of labeling functions (LFs)
lfs=[lf_keyword_conference,lf_keyword_ene_ect,lf_keyword_management,lf_keyw
ord_personal,lf_keyword_project,lf_keyword_resumes,lf_keyword_universities]
# Apply the LFs to the unlabeled training data
applier=PandasLFApplier(lfs=lfs)
L_train=applier.apply(df=train)
L_test = applier.apply(df=test)
```

Step 7 - Training Model

- **Predict:** Return predicted labels, with ties broken according to policy.
- **Fit:** Train majority class model.
- **Predict_proba:** Predict probabilities using majority vote.

```
majority_model=MajorityLabelVoter(cardinality=5)
preds_train=majority_model.predict(L=L_train)
```

```
label_model=LabelModel(cardinality=5,verbose=True)
label_model.fit(L_train=L_train, n_epochs=500, log_freq=100,seed=123)
```

```
probs_train = label_model.predict_proba(L=L_train)
```

Step 7 - Training Model

```
from snorkel.labeling import filter_unlabeled_dataframe

df_train_filtered, probs_train_filtered =

filter_unlabeled_dataframe(X=train, y=probs_train, L=L_train)
```

Step 8 - Testing Model

- **CountVectorizer:** Convert a collection of text documents to a matrix of token counts.
- **Transform:** The transform method is transforming all the features using the respective mean and variance.
- **Fit:** The fit method is calculating the mean and variance of each of the features present in our data.

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(ngram_range=(1, 5))
X_train = vectorizer.fit_transform(df_train_filtered.XFolder.tolist())
X_test = vectorizer.transform(test.XFolder.tolist())
```

Step 8 - Testing Model

Probs_to_preds: Convert an array of probabilistic labels into an array of predictions.

```
from snorkel.utils import probs_to_preds

preds_train_filtered = probs_to_preds(probs=probs_train_filtered)
```

Step 8 - Testing Model

```
from sklearn.linear_model import LogisticRegression
sklearn_model = LogisticRegression(C=1e3, solver="liblinear")
sklearn_model.fit(X=X_train, y=preds_train_filtered)
```

```
print(f"Test Accuracy: {sklearn_model.score(X=X_test, y=Y_test) *
100:.1f}%")
```

```
Test Accuracy: 69.1%
```

Thanks You

LIANG, Ting-Yu