Gender classification

Objective

Given the first name of a person, predict the gender.

Exploratory Data analysis

Let's dive into the data!

---

1. Overall statistics of the data

- Total population count - 514,771,254
  (*** i.e., customers whose gender was tagged as 'MALE' or 'FEMALE')
- Total unique combination of ***first name*** and ***gender*** that constitutes to above population – 33,285,798

2. Analyzing first name

Observed following data inconsistencies in the first name column:

- Names with length less than 3 letter like a, aa, ab, xx etc.
- Names entered as cust-01929920929
- First name column had middle name and last name Mohan lal shah
- Garbage or special characters as & @, . - , xxxxx,zzzzz ,---- etc

Exploratory Data analysis...contd

3. Cleaning first name

- Total population count – 484,346,718
  (*** Total cleansed firstname population count #ignore 'noname')
- Total unique combination of ***cleansed first name*** and ***gender*** that constitutes to above population – 27,648,771

Logic to clean first name

```
if len(name)<3 or name[:4]=='cust':
        name='noname'

    #split the name , sometime firstname column also has surname
    name=name.split()[0]

    #remove occurences xxxxx,zzzzz,---,.....,@@ etc
    name=''.join(re.findall('[x|z|-]*([a-zA-Z]+?)[z|x|-]*',name))

    if not name.isalpha():
        name='noname'

    if len(name)<3:
        name='noname'
```

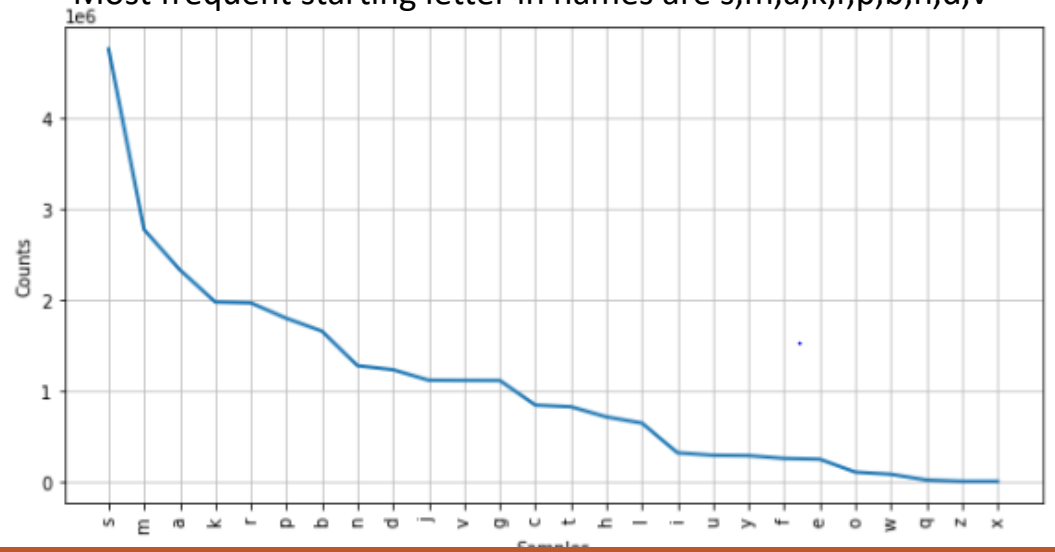
Exploratory Data analysis…contd

4. For every first alphabet of name(cleansed), top 2000 most frequent names

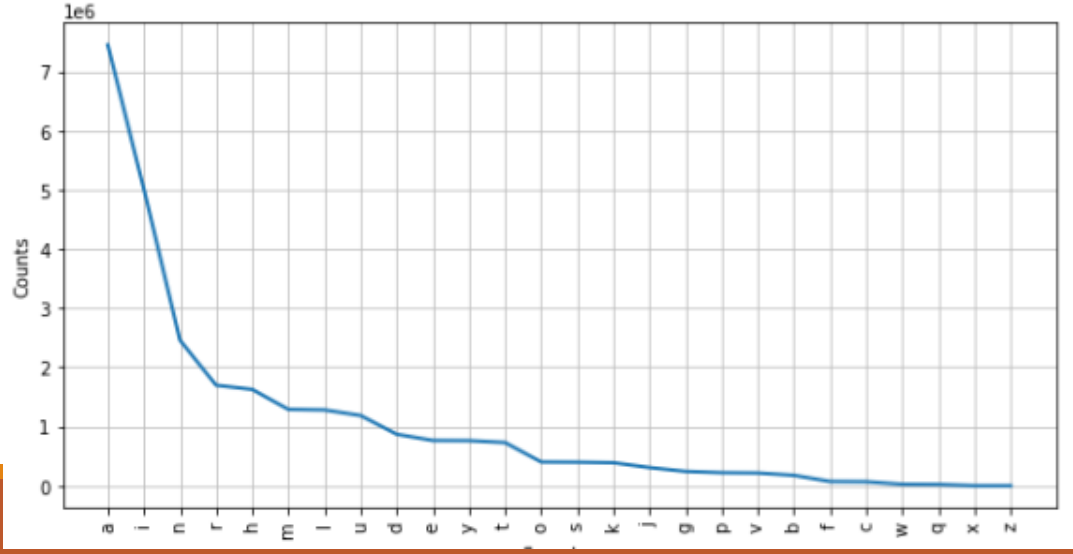
- Total population count – 406,096,983
  (*** Total cleansed firstname population count #ignore 'noname')
- Total unique combination of ***cleansed first name*** and ***gender*** that constitutes to above population – 48,196
- We are able to capture 79 % population,and the combination has reduced from 2.76 cr to 48k. Good size for training
- Check more details in this excel - Gender EDA.xlsx

5. Analyze first letter and last letter of name

Most frequent starting letter in names are s,m,a,k,r,p,b,n,d,v



Most frequent last letter in names are a,i,n,r,h,m,l,u,d,e,y



Exploratory Data analysis…contd

6. Training data has both genders for the same name

To get rid of this, consider the most frequent gender for a particular name:

- Total population count – 365,650,956
- Total unique combination of ***cleansed first name*** and ***top gender*** that constitutes to above population – 48,196
- We were able to capture 71 % populations, and the combination is still 48k.
- Check more details in this excel - Gender EDA.xlsx

7. Final Training data

Gender	Training count
FEMALE	18219
MALE	29977
Grand Total	**48196**

***Data is imbalanced, we might need to resample before feeding into the model*

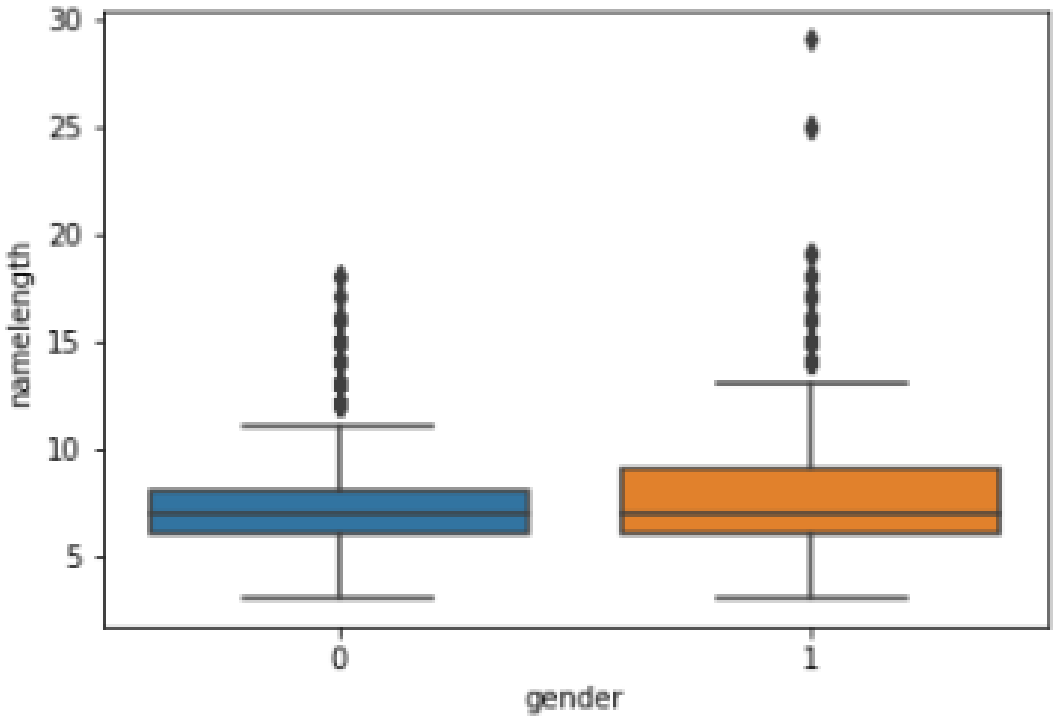
Feature engineering

In order to identify gender of a person , what could be important parameters?

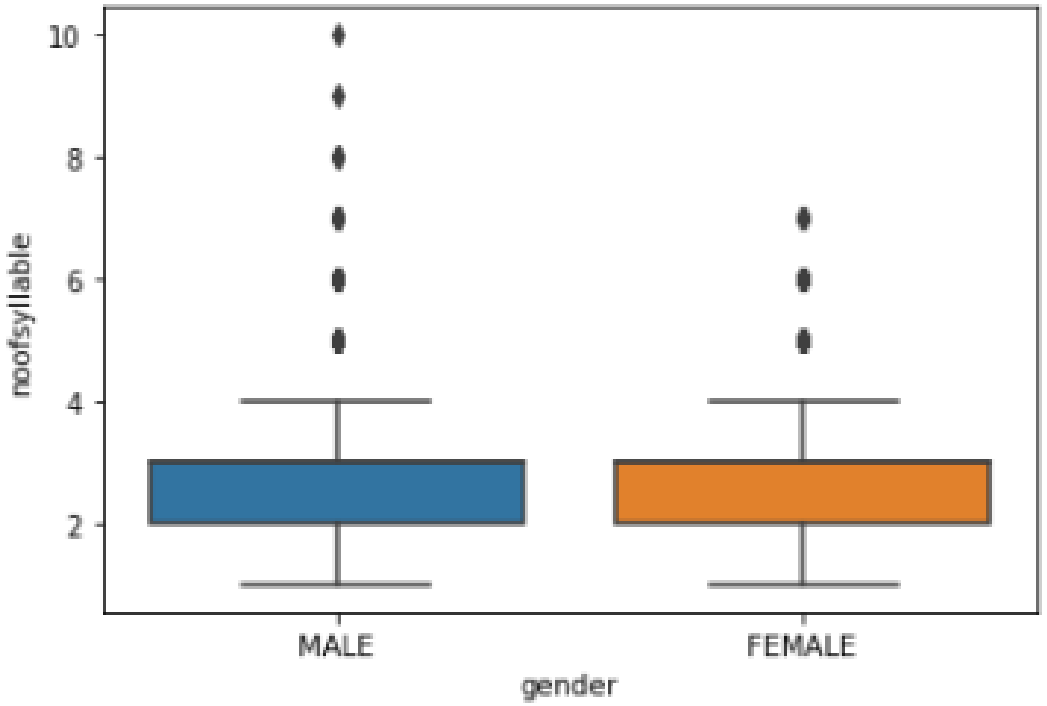
- Pronunciation
- Length of Name
- Vowels and its position
- First and last character of name
- Combination of first and last characters in a name
- Stressed phonetics
- Metaphones

Feature engineering ..contd

Length of name

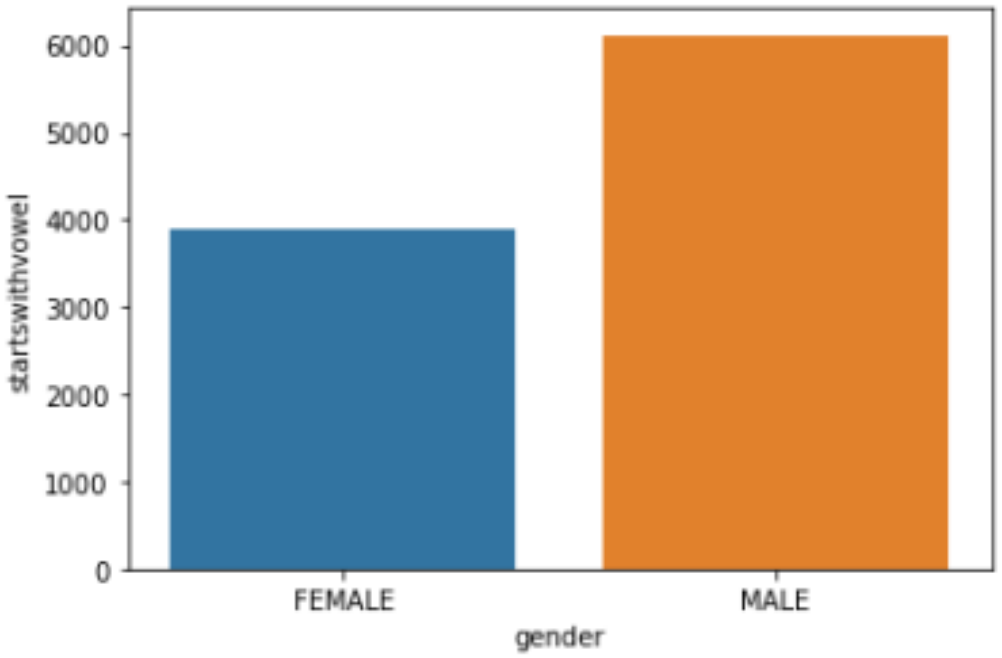


No of syllable

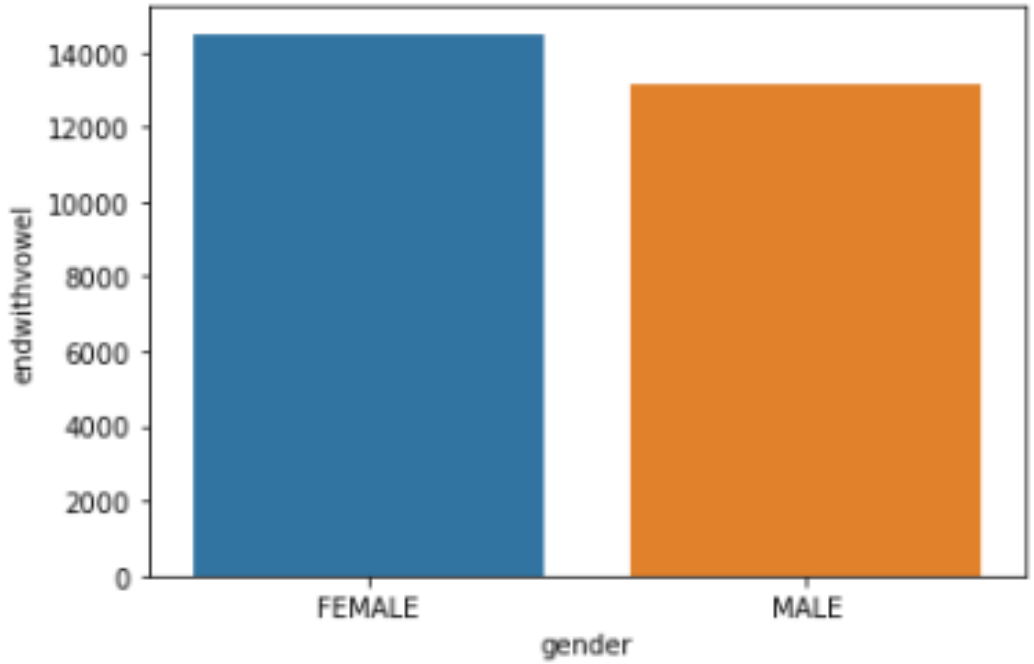


Feature engineering ..contd

Starts with Vowel



Ends with Vowel



Feature engineering ..contd

Metaphone

Extracted total 12616 distinct metaphone (a type of algorithm which identifies similar-sounding words)

Female Probability	unique metaphone
< 50 %	8,595
>=50 % and < 80%	652
>=80%	3,368
Grand Total	**12,615**

Male Probability	unique metaphone
< 50 %	4,013
>=50% and < 80 %	723
>=80 %	7,879
Grand Total	**12,615**

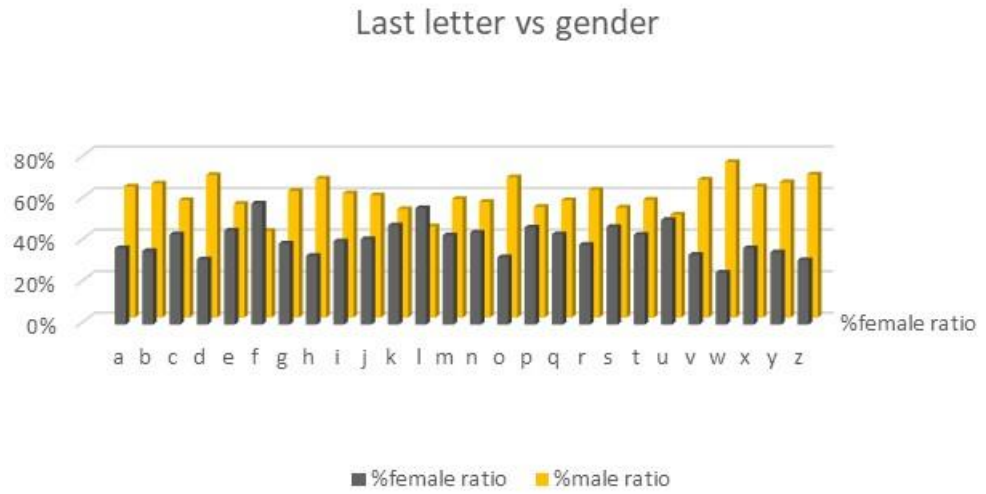
Note : 3368 can help us identify female and 7879 metaphone can help us identify males with 80 % accuracy

Refer excel for more detail metaphone.CSV

Feature engineering ..contd

First letter of name

Last letter of name

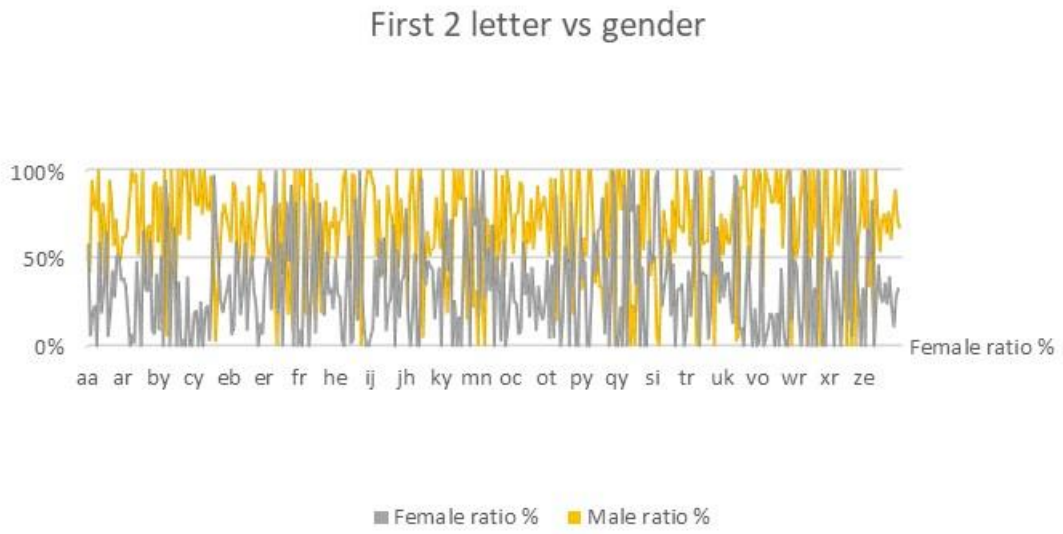


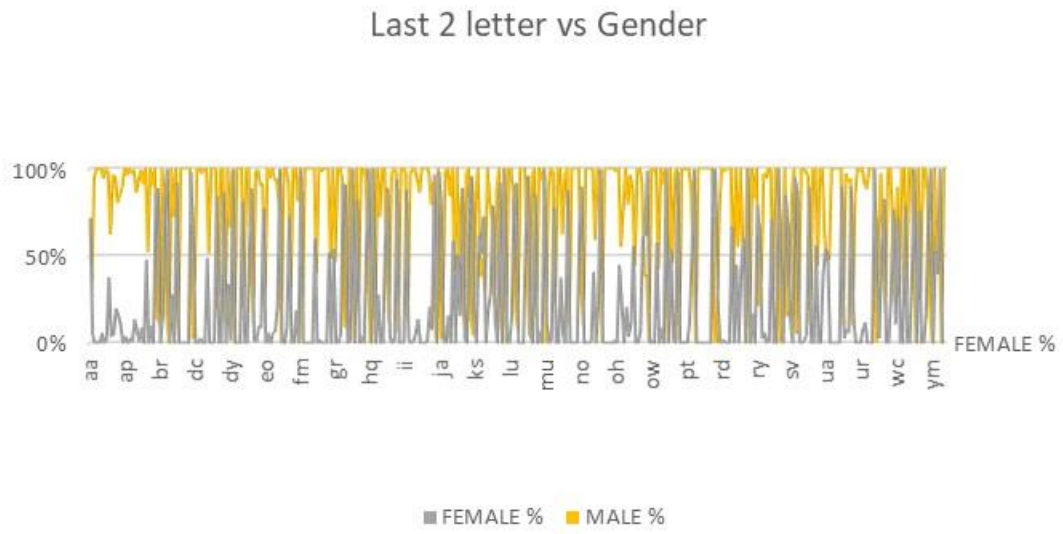
Only first letter and last letter does not help much, as it is mostly tilted towards Male

Feature engineering ..contd

First 2 letter of name

Last 2 letter of name



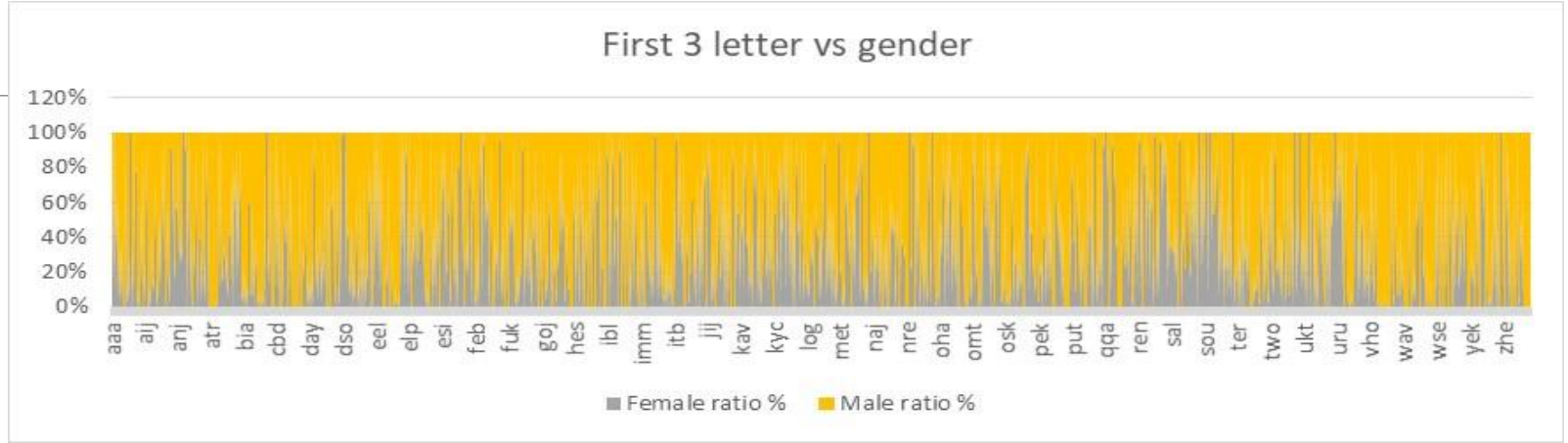


First 2 and last 2 letters does a better job in classifying the gender

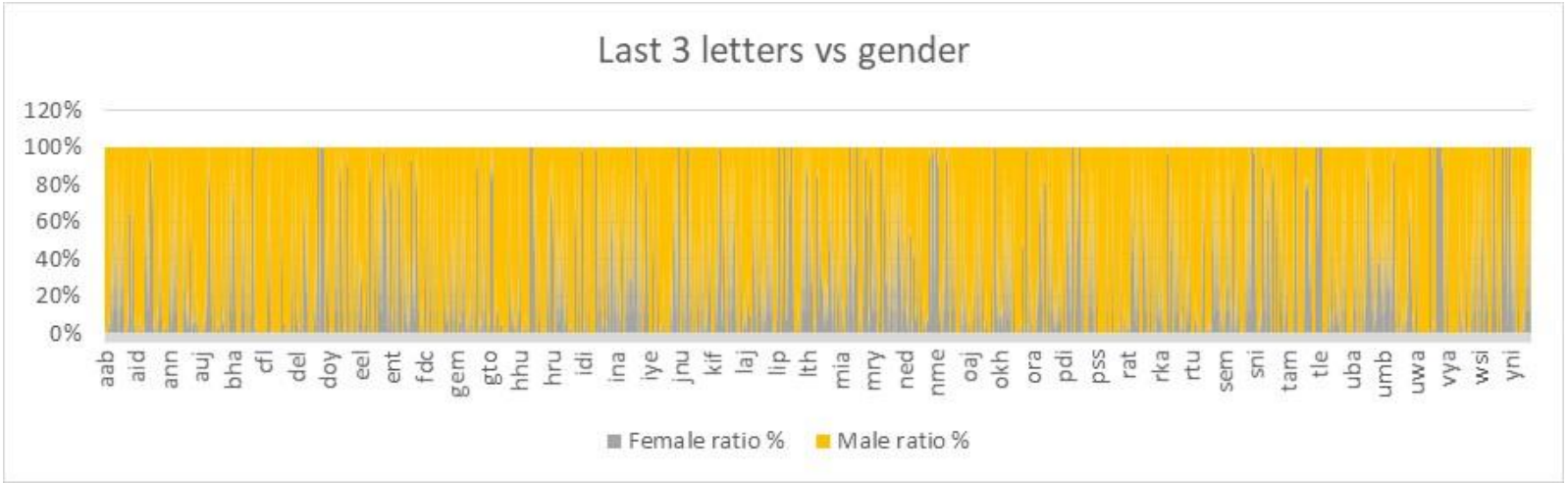
Feature engineering ..contd

First-3 & Last-3 letters of name

- Out of the 2999 total distinct combination of last-3 letters
  - 590 combinations can classify name as female in 80% of accuracy
  - 1411 combinations can classify name as male in with 80% accuracy



- Out of the 2665 total distinct combination of last-3 letters
  - 665 combinations can classify name as female in 80% of accuracy
  - 1611 combinations can classify name as male in with 80% accuracy



Building ML models

Type	Metric	Logistic regression	Random Forest (100)	xgboost(20)
Original training data	accuracy	0.84	0.86	0.83
	f1 score	0.82	0.85	0.82
	auc score	0.90	0.93	0.91
Resampled data	accuracy	0.86	0.94	0.84
	f1 score	0.86	0.94	0.83
	auc score	0.91	0.98	0.92

Deploying model in production

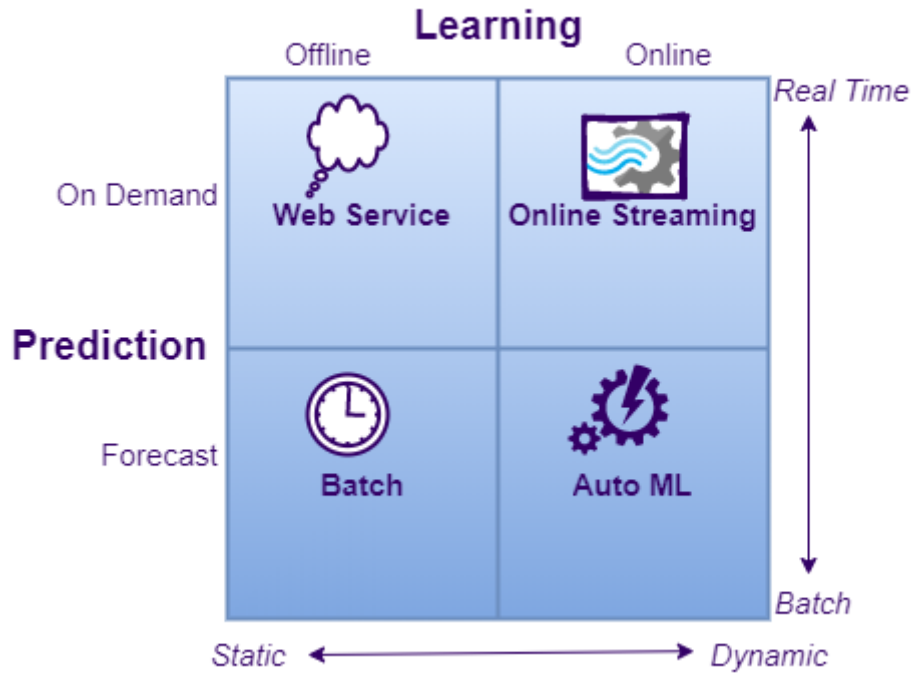
ML Pipeline

To ensure minimal human intervention and pre-processing dependencies of models, following steps needs to be handled during production deployment:

- Cleaning
- Feature extraction
- Label encoding
- Vectorizing features
- Any other column transformation or customized function transformation

To do that, python has "pipeline" feature for streamlining your model deployment into production. End user don't have to worry about the preprocessing steps-as all the cleaning, transformations, feature extraction can be packaged in pipeline and saved as a pickle file.

Deployment methods



Deploying model in production..contd

Below is a code snippet which I used to construct the pipeline for “gender classification” model.

```
from sklearn.pipeline import Pipeline,FeatureUnion
from joblib  import dump,load
```

```
def main():
    data=connection()
    x_train,x_test,y_train,y_test=gettrainingdata(data)

    pipeline=Pipeline(steps=[
                    ('cleaning',CleanName()),
                    ('features',NameFeature()),
                    ('dictvectorizer',DictVectorizer()),
                    ('model',RandomForestClassifier(n_estimators=100))])

    pipeline.fit(x_train,y_train)
    getRoccurve(y_test,pipeline.predict_proba(x_test)[:,1])
    print(classification_report(y_test,pipeline.predict(x_test)))

    dump(pipeline,filename="gender_classification.joblib")
```

- Created Class “CleanName” and “NameFeature” to clean the name passed to model and extract the features
- Note: We could have just created the “cleanname()” and “namefeature()” to handle to transformation along with pipeline.FunctionTransformation. However, pipeline needs to serialize the transformation functions as an object, so it can be fetched as an object during prediction.
- To know more about pipeline, refer to the below link
- https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html

```
from joblib  import dump,load
pipe=load("gender_classification.joblib")

pipe.predict(['Amit','Ramesh','Amresh','Vandana'])
```

```
array([1, 1, 1, 0])
```

Deploying model in production..contd

```
from sklearn.base import BaseEstimator,TransformerMixin
import syllables
import phonetics
import re
import numpy

class NameFeature(BaseEstimator,TransformerMixin):
    def __init__(self):
    def fit(self,X,y=None):
    def transform(self,X):
    def features(self,X):
        return ({
            'first-letter':X[0],
            'first-2letter':X[:2],
            'first-3letter':X[:3],
            'last-letter':X[-1],
            'last-2letter':X[-2:],
            'last-3letter':X[-3:],
            'namelength':len(X),
            'startswithvowel':1 if X[0] in 'aeiou' else 0,
            'endwithvowel':1 if X[-1] in 'aeiou' else 0,
            'noofsyllable':syllables.estimate(X),
            #'soundex':phonetics.soundex(name)
            'metaphone':phonetics.metaphone(X)
            #'nys':ph.nysiis(name)
            #'count':cnt
        })

class CleanName(BaseEstimator,TransformerMixin):
    def __init__(self):

    def fit(self,X,y=None):

    def transform(self,X):

    def getfirstname(self,X):
        #remove names starting with 'cust', cust-019292020 etc
        try:
```

**Creating your own module to handle data preprocessing

Deploying model in production..contd

Basic flask deployment

```
from flask import Flask, render_template, request, redirect, url_for
from joblib  import dump,load


app = Flask(__name__,template_folder='./templates')

pipe=load("gender_classification.joblib")

def requestResults(name):
    predicted_gender = pipe.predict(name.split())

    return "Gender is "+ str(predicted_gender)

@app.route('/')
def home():
    return render_template('home.html')

@app.route('/', methods=['POST', 'GET'])
def get_data():
    if request.method == 'POST':
        user = request.form['search']
        return redirect(url_for('success', name=user))


@app.route('/success/<name>')
def success(name):
    return "<xmp>" + str(requestResults(name)) + " </xmp> "

if __name__ == '__main__' :
    app.run(port=5002,debug=True)
```

1. Load model

4. Predict using model

2. Create html page

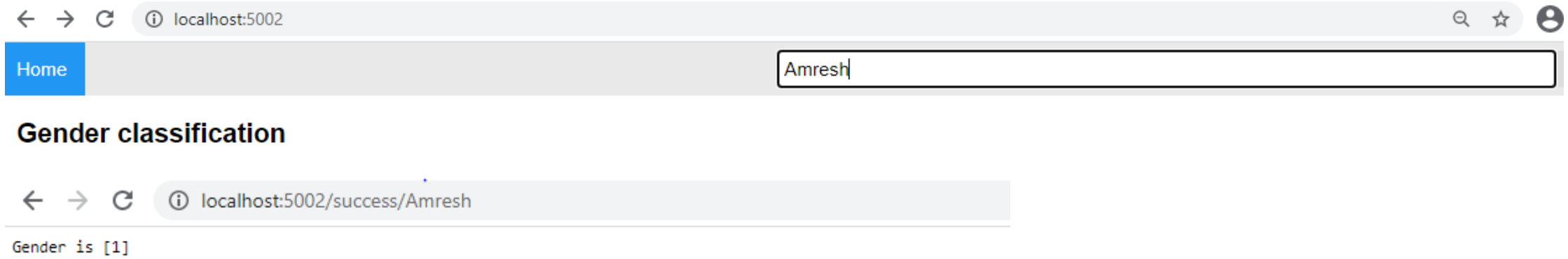
3. Get input

5. Return prediction for display

Deploying model in production..contd

Web deployment result

Input sample 1



Input sample 2