

Language Detection

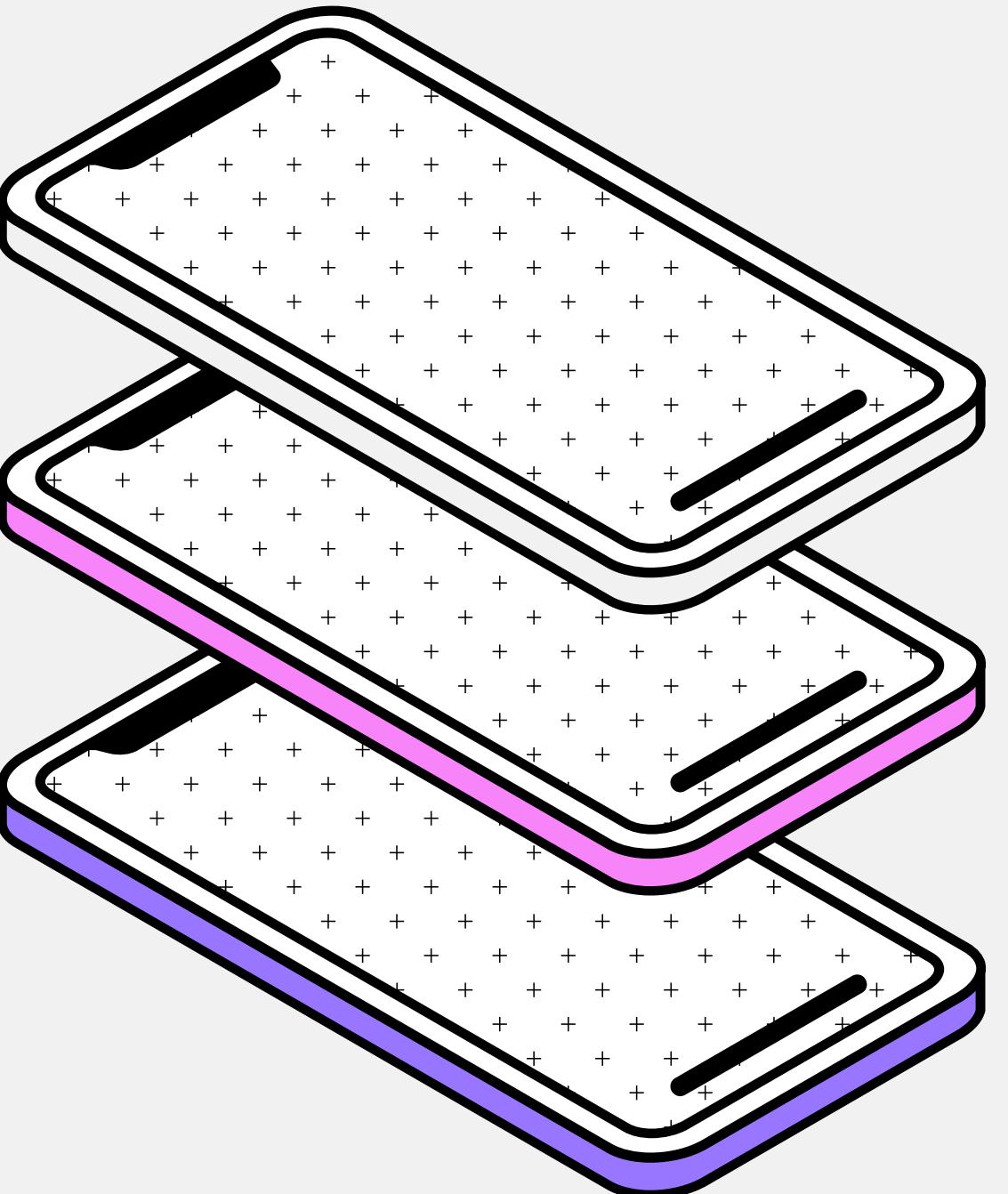
WITH MULTINOMIAL NAIVE BAYES

Created by

2440004445 - Steven Matthew

2401860092 - Samuel Chandra

2440050874 - Michael Julian



Why Language Detection?

Kami ingin membuat model yang dapat membedakan berbagai jenis bahasa terutama bahasa dengan aksara sejenis, juga aksara yang unik seperti bahasa Mandarin, Korea, Jepang, Hindi, dan lainnya.

Pembuatan model ini dapat membantu kami untuk mendekripsi bahasa yang kita temui sehari-hari seperti di email, chat, sosial media, dan lain-lain.



Dataset

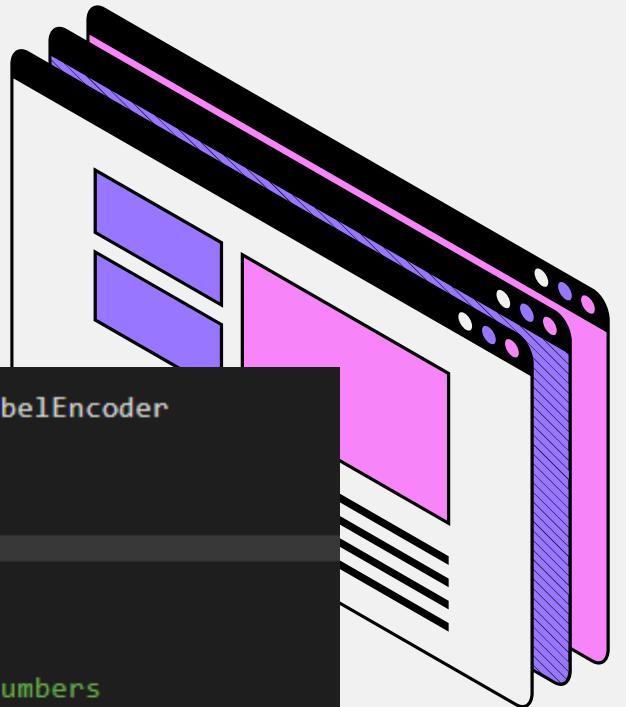
21 Languages

11.000 Rows of Sents.

2 kaggle

0	oyama station was opened on april the statio...	English
1	elle est la fille de gérios tuéni et katbé sur...	French
2	en participó en el video "love machine" donde...	Spanish
3	o rondonópolis e o goiás são os únicos times d...	Portuguese
4	immagino che non vorrebbe più pane d'oro adess...	Italian
5	мая года был назначен тренером вратарей сбор...	Russian
6	cerro wachu kkollu är ett berg i bolivia det l...	Swedish
7	ടെറി നിങ്ങൾ യമാർത്തമത്തിൽ ആ മാലാവയേപ്പാലെയാണ് ക...	Malayalam
8	francisco de miranda is een gemeente in de ven...	Dutch
9	في أبريل مقدم الرعيم مصطفى النحاس مذكرة للسف...	Arabic
10	peugeot aslinda peugeot efsane kuşağıının ürünü...	Turkish
11	Ihr zwei seid so nett	German
12	டெர்ரி நீங்கள் உண்மையில் அந்த தேவதையைப் போலவே ...	Tamil
13	terry du ser faktisk lidt ud som den engel, me...	Danish
14	ಟರீ நீங்கள் நிஜவாயில் அந்த தேவதையைப் போலவே ...	Kannada
15	Terry, στην πραγματικότητα μοιάζεις λίγο με αυ...	Greek
16	यी की अपनी चित्रलिपि पर आधारित एक लिपि है जिसम...	Hindi
17	在演说中，他呼吁全人类团结起来，共同反对专制、贫困、疾病和战争，他在演说中提到的：“不要问你...	Chinese

Data Preprocessing



Label Encoder

encoding
target to labels

Symbols & Numbers

removing
symbols and
numbers from
features

Lower Case

lower casing
features

```
[ ] from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)

[ ] data_list = []
for x in X:
    # removing the symbols and numbers
    x = re.sub(r'[@#$(),%^*?:;~0-9]', ' ', x)
    x = re.sub(r'[[[]]]', ' ', x)
    # converting the text to lower case
    x = x.lower()
    #remove mentions
    x = re.sub('@\S+', ' ', x)
    #remove url
    x = re.sub('https*\S+', ' ', x)
    #remove hashtag
    x= re.sub('#\S+', ' ', x)
    # appending to data_list
    data_list.append(x)
```

Mentions

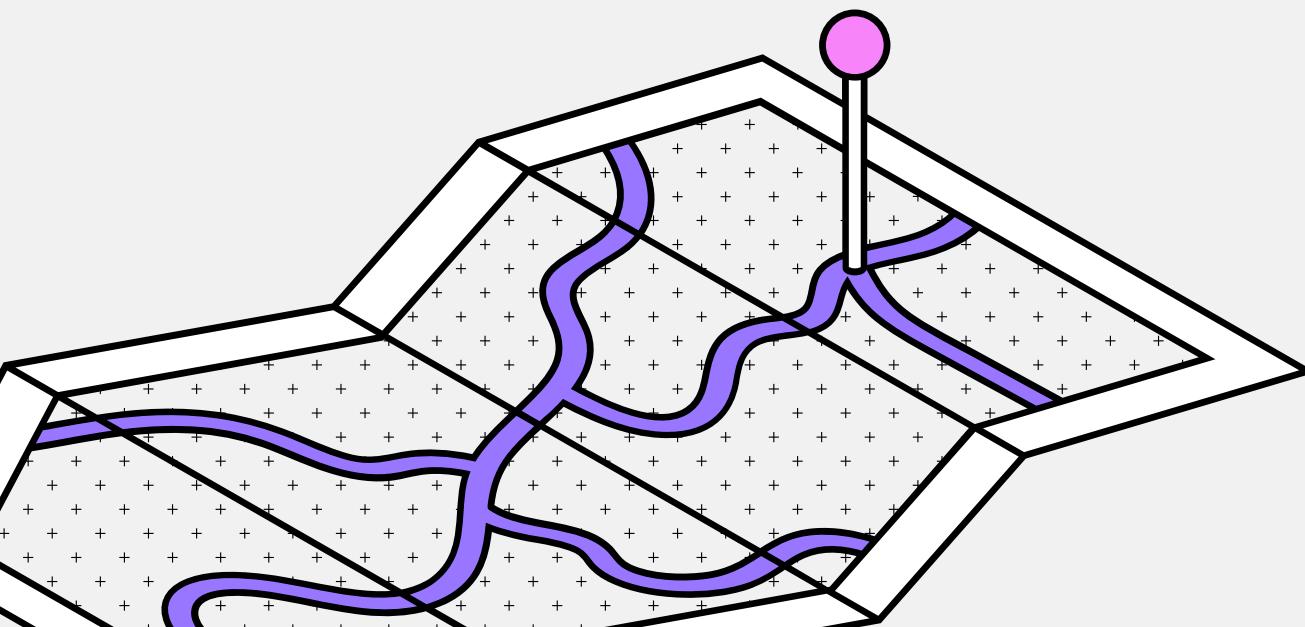
removing
mentions

URL

removing URLs

Hashtag

removing
hashtags

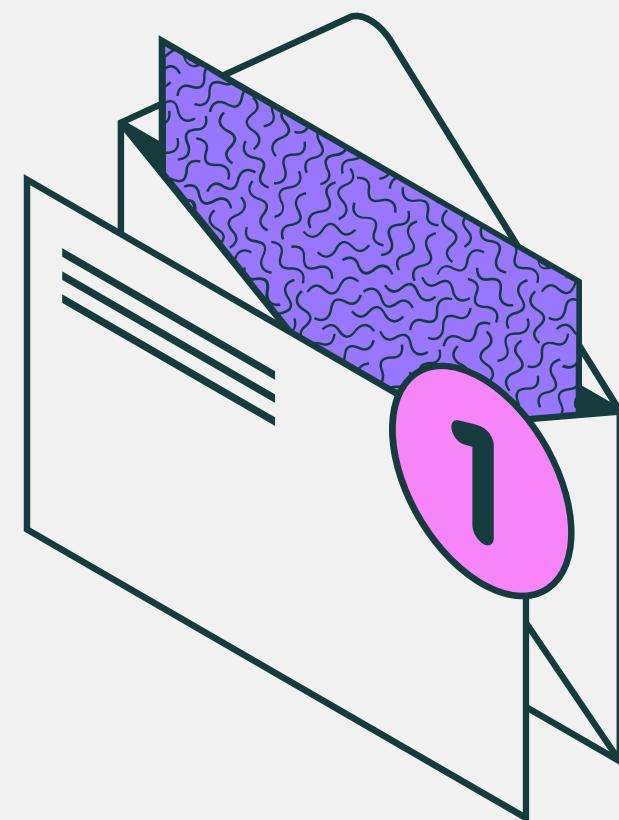


Bag of Words (BoW)

Feature Text Vectorization

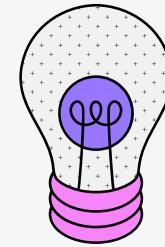
using **CountVectorizer**

```
[ ] from sklearn.feature_extraction.text import CountVectorizer  
cv = CountVectorizer()  
X = cv.fit_transform(data_list).toarray()  
X.shape  
  
(11000, 58874)
```



Data Splitting

train_test_split



Ratio

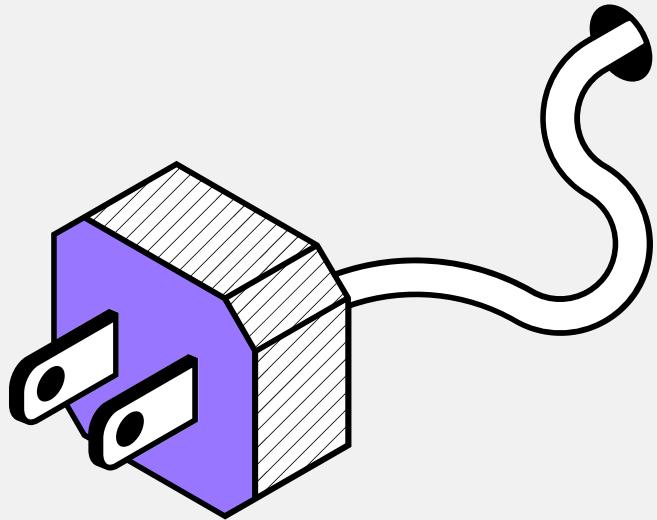
Train : Test

(4 : 1)

(8800 : 2200)

```
[ ] from sklearn.model_selection import train_test_split  
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 42)
```

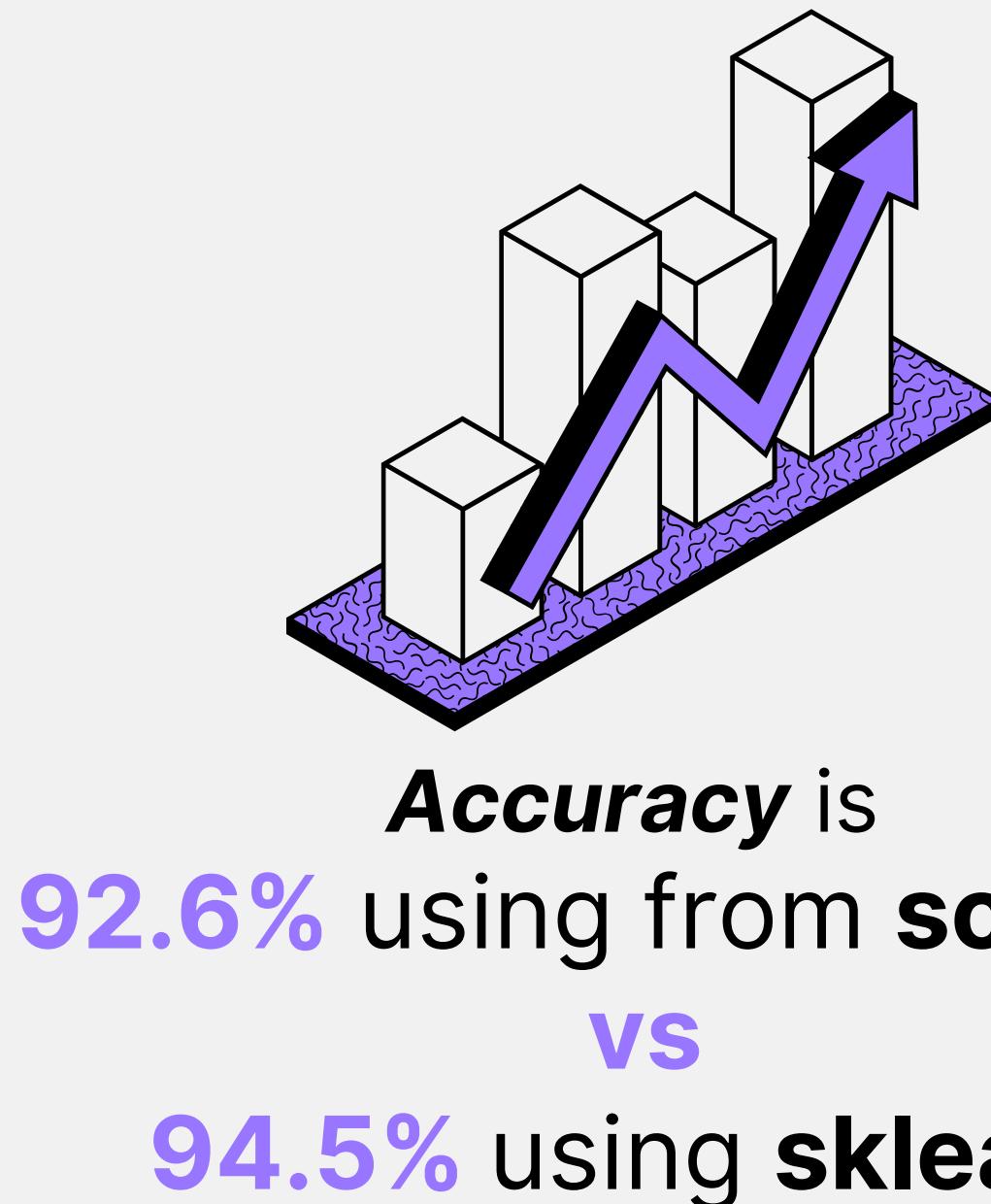
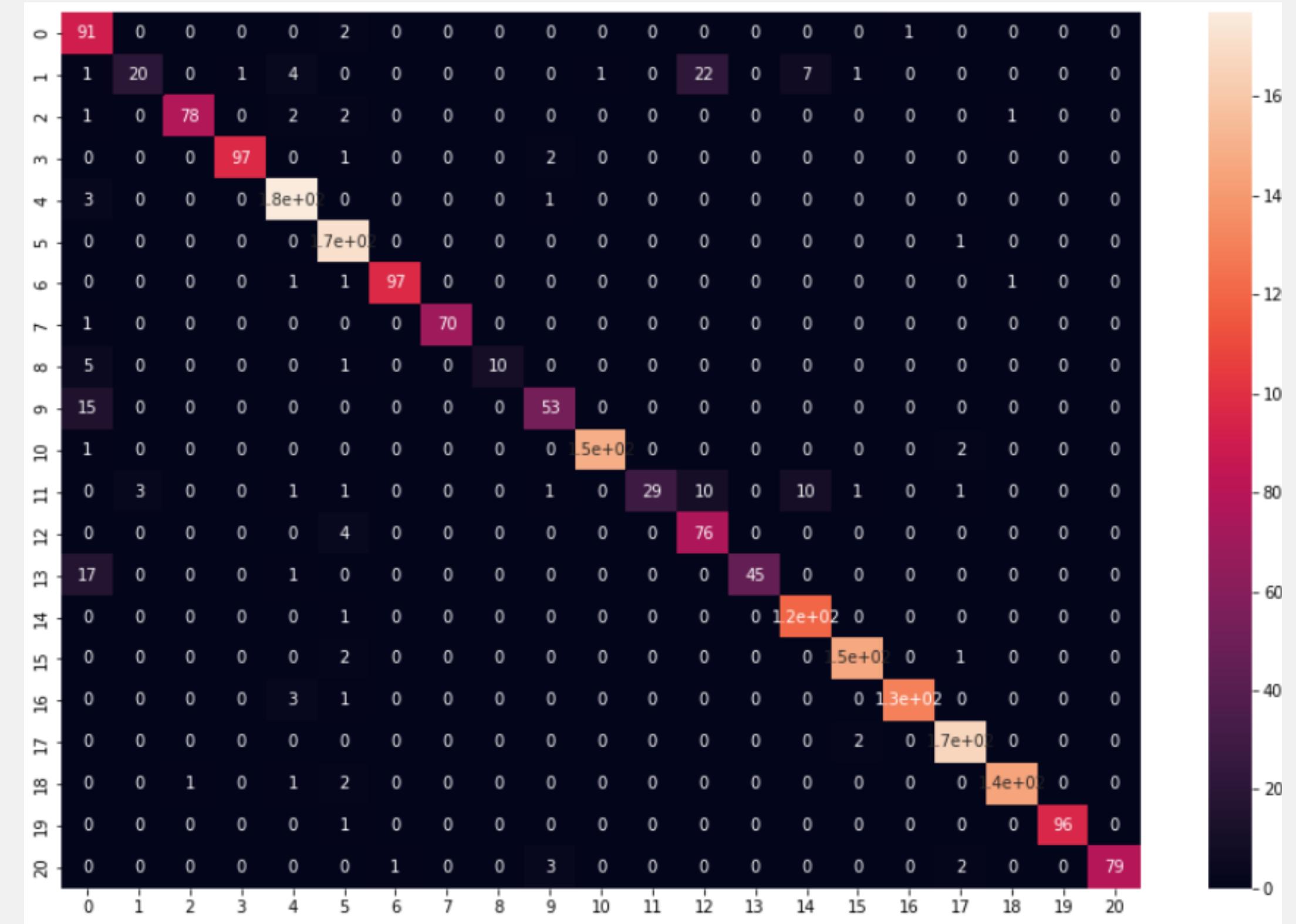
Building Multinomial Naive Bayes Model from scratch



```
[ ] class MultinomialNB:  
    def __init__(self,alpha=1):  
        self.alpha = alpha  
  
    def _prior(self):  
        P = np.zeros((self.n_classes_))  
        _, self.dist = np.unique(self.y,return_counts=True)  
        for i in range(self.classes_.shape[0]):  
            P[i] = self.dist[i] / self.n_samples  
        return P  
  
    def fit(self, X, y):  
        self.X = X  
        self.y = y  
        self.n_samples, self.n_features = X.shape  
        self.classes_ = np.unique(y)  
        self.n_classes_ = self.classes_.shape[0]  
        self.class_priors_ = self._prior()  
  
        self.uniques = []  
        for i in range(self.n_features):  
            tmp = np.unique(X[:,i])  
            self.uniques.append( tmp )  
  
        self.N_yi = np.zeros((self.n_classes_, self.n_features))  
        self.N_y = np.zeros((self.n_classes_))  
        for i in self.classes_:  
            indices = np.argwhere(self.y==i).flatten()  
            columnwise_sum = []  
            for j in range(self.n_features):  
                columnwise_sum.append(np.sum(X[indices,j]))  
  
            self.N_yi[i] = columnwise_sum  
            self.N_y[i] = np.sum(columnwise_sum)  
  
    def _theta(self, x_i, i, h):  
        Nyi = self.N_yi[h,i]  
        Ny = self.N_y[h]  
  
        numerator = Nyi + self.alpha  
        denominator = Ny + (self.alpha * self.n_features)  
  
        return (numerator / denominator)**x_i  
  
    def _likelyhood(self, x, h):  
        tmp = []  
        for i in range(x.shape[0]):  
            tmp.append(self._theta(x[i], i,h))  
  
        return np.prod(tmp)  
  
    def predict(self, X):  
        samples, features = X.shape  
        self.predict_proba = np.zeros((samples,self.n_classes_))  
  
        for i in range(X.shape[0]):  
            joint_likelyhood = np.zeros((self.n_classes_))  
  
            for h in range(self.n_classes_):  
                joint_likelyhood[h] = self.class_priors_[h] * self._likelyhood(X[i],h)  
  
            denominator = np.sum(joint_likelyhood)  
  
            for h in range(self.n_classes_):  
                numerator = joint_likelyhood[h]  
                self.predict_proba[i,h] = (numerator / denominator)  
  
        indices = np.argmax(self.predict_proba,axis=1)  
        return self.classes_[indices]
```

Evaluation

Confusion Matrix

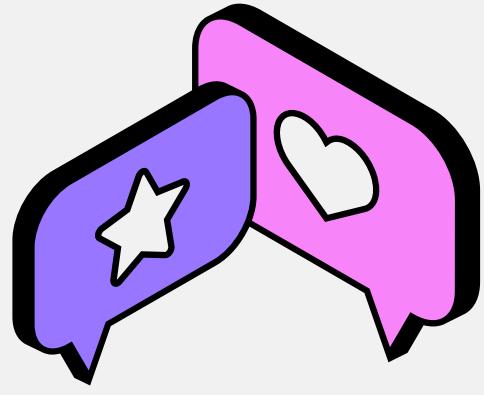


Code

https://colab.research.google.com/drive/1u5iP0tGa-B3AeF6f1iSTUH_3e_9fYKTX#scrollTo=ivWGIdxs5v6U

References

<https://www.kaggle.com/code/ayushmi77ai/language-detection-nlp>



Thank you