

## Exercise 3

# Blog Gender Classification

---

Sunil Gauda

ID : 10595858



## Table Of Contents

<b>Table Of Contents</b>	<b>1</b>
<b>Business Understanding</b>	<b>2</b>
<b>Data Understanding</b>	<b>2</b>
<b>Data Preparation</b>	<b>2</b>
Numerical Output	2
Null Management	3
<b>Modeling</b>	<b>5</b>
<b>Evaluation</b>	<b>5</b>
Classification Report	6
Confusion Matrix	6
<b>Deployment</b>	<b>7</b>
Using Pickle	7
<b>RapidMiner Automodel Process</b>	<b>8</b>
Uploading Data	8
Selecting the Column and the Task we need to perform	9
Overview	9
Input Selection	9
Model Selection	10
Results	11

## Business Understanding

Gender identification is majorly used to check the dynamics of the text which can be used to classify the text as per gender. The classification can form a basis for identifying needs regarding gender and can help us analyze the behavior of the user.

## Data Understanding

	BLOG	GENDER
0	Beyond Getting There: What Travel Days Show U...	F
1	I remember so much about the island; the large...	F
2	I have had asthma and allergies my entire life...	M
3	The last few days have been an emotional rolle...	M
4	If you lined up all the teachers and staff in ...	F

fig 1 Data

The Data above comprises Two Columns, BLOG and GENDER, the data contains the necessary information for developing a classification system. By Processing the data using tokenization, lemmatization, and vectorisation we can create a model to predict gender from text.

## Data Preparation

### Numerical Output

As the classification is just binary, The column Gender can be converted to just 1, 0 for M and F representing Male and Female. This will help us computation time and prevent errors on model training as the output data is numerical.

Conversion of data of GENDER column to 1 and 0 as it is a binary classification

```
for gen in df['GENDER']:
    if gen=='M':
        df['GENDER'].replace({'M': '1'}, inplace=True)
    elif gen=='F':
        df['GENDER'].replace({'F': '0'}, inplace=True)
```

✓ 0.1s

fig 2 Data Modification

## Null Management

```
df.dropna(inplace=True)
```

✓ 0.1s

fig 3 Null Value Management

Dropping null columns to prevent errors due to invalid data, the data was already blanched and cleared, but as a caution the null columns need to be dropped, as the data comprises of text as input params we cannot generate or augment the data in anyway, hence we need to drop the columns of the data.

## Tokenize The Data

Entire Sentence does not produce any viable meaning to a machine learning algorithm, so we need to split the data to each word and create smaller units that we can work with.

```
df["BLOG_Token"] = [word_tokenize(post) for post in df['BLOG']]
✓ 21.7s
```

```
df.head()
```

	BLOG	GENDER	BLOG_Token
0	Beyond Getting There: What Travel Days Show U...	0	[Beyond, Getting, There, :, What, Travel, Days...
1	I remember so much about the island; the large...	0	[I, remember, so, much, about, the, island, ;,...
2	I have had asthma and allergies my entire life...	1	[I, have, had, asthma, and, allergies, my, ent...
3	The last few days have been an emotional rolle...	1	[The, last, few, days, have, been, an, emotion...
4	If you lined up all the teachers and staff in ...	0	[If, you, lined, up, all, the, teachers, and, ...

fig 4 Tokenizing

## Lemmatization

The problem in text data is repetition of the words, It creates ambiguity to the meaning of the word, using lemmatization we can reduce the ambiguity by grouping similar words together, the challenge is to get reference data for the lemmatizer to work, WordNetLemmatizer is one such library that we can use to compare and group the words in a sentence.

```
desc_new_lem = []
lem = WordNetLemmatizer()
for each_row in post_new_alpha:
    desc_new_lem.append([lem.lemmatize(word) for word in each_row])

df["BLOG_Token_cleaned"] = desc_new_lem
df["BLOG_Token_cleaned"] = [" ".join(desc) for desc in df['BLOG_Token'].values]
✓ 6.7s
```

fig 5 Lemmatization using WordNetLemmatizer

## Vectorising

As a computer does not understand textual data, we use vectorisation to convert textual processed data to numerical data, we have used TFIDFVectoriser as we need to only focus on frequency of the occurrence of the words in the textual data.



```
Vectorising

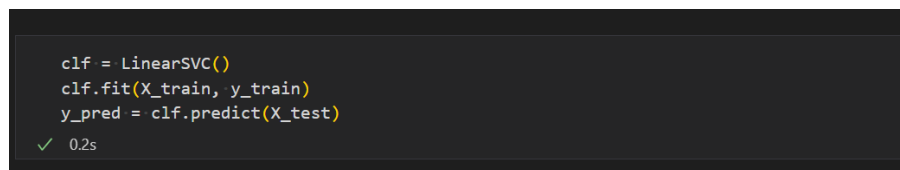
tft = TfidfVectorizer(max_features=1000)
x = tft.fit_transform(df["BLOG_Token_cleaned"]).toarray()
X = pd.DataFrame(x)

2] ✓ 2.6s
```

fig 6 TFIDF Vectorising

## Modeling

Modeling comprises selection of appropriate machine learning algorithms to train on our data. LinearSVC Performs better on small datasets.



```
clf = LinearSVC()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

✓ 0.2s
```

fig 7 LinerSVC Modelling

## Evaluation

We need to Evaluate the model to check its accuracy, in classification we have different means of validating the results out of which, Classification Report and Confusion Matrix is used.

## Classification Report

	precision	recall	f1-score	support
0	0.66	0.65	0.65	252
1	0.67	0.68	0.68	268
accuracy			0.67	520
macro avg	0.67	0.66	0.66	520
weighted avg	0.67	0.67	0.67	520

fig 8 Classification Report

The model gives the accuracy of 67%, and precision of 66% with recall of 65%.

## Confusion Matrix

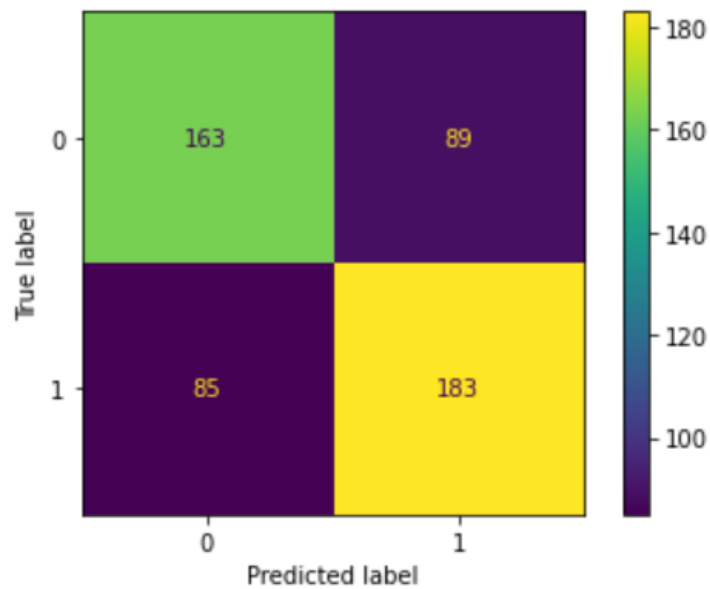


fig 9 Confusion Matrix

## Deployment

### Using Pickle

We can use Built In module called pickle from python to store and load the model to predict the outcome, but the data processing is needed before input is provided to the pickled model.

```
pk1.dump(clf, open("clfModel.pkl", 'wb'))  
✓ 0.2s
```

fig 10 Saving the Model



## RapidMiner Automodel Process

Rapid miner is a platform that does all the processes above with ease and simplicity, it applies the models and provides us the results and has an entire pipeline to do rapid machine learning development.

### Uploading Data

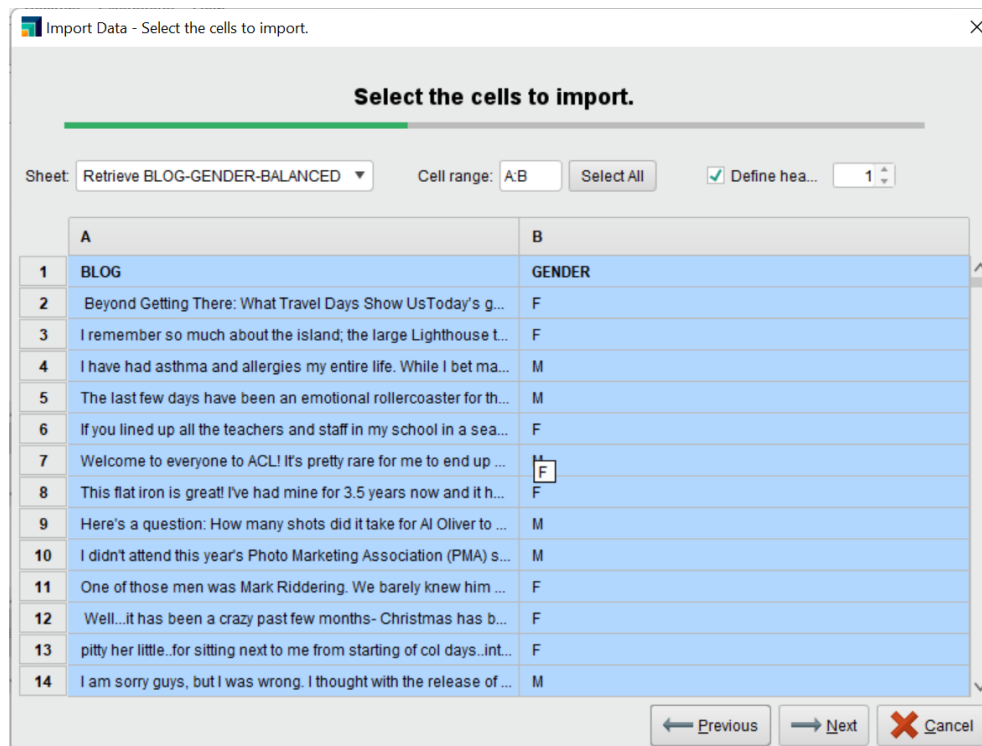


fig 11 Importing Data

We have to select the File from our system to upload it to rapid-miner.

## Selecting the Column and the Task we need to perform

We select, predict and select the column GENDER as it is our target variable.

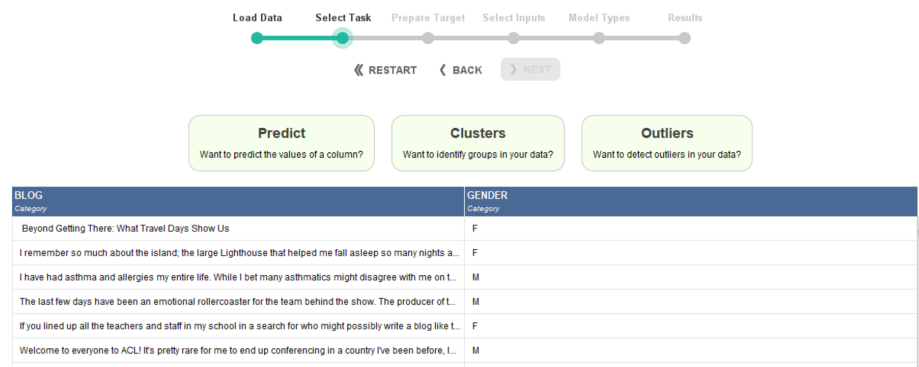


fig 12 Selecting Task and Output Column

## Overview

Following is the overview of the data for output variable

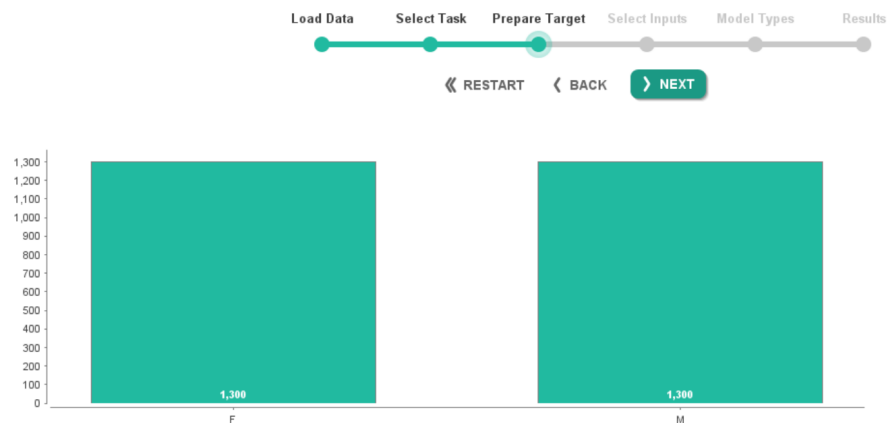


fig 13 Overview

## Input Selection

We will select the only input displayed as its a binary classification

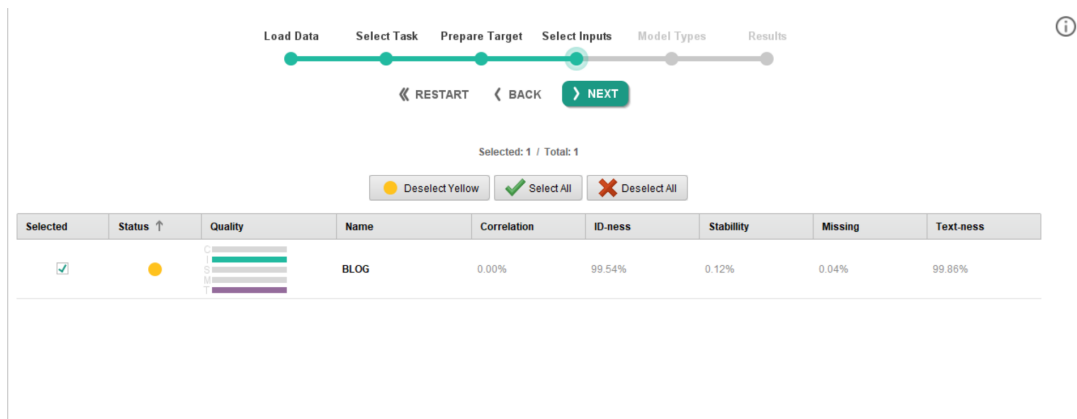


fig 14 Input Selection

## Model Selection

Select what machine learning models we need to apply to our data, the models are implemented by rapid miner and parameters are adjusted by rapid miner.

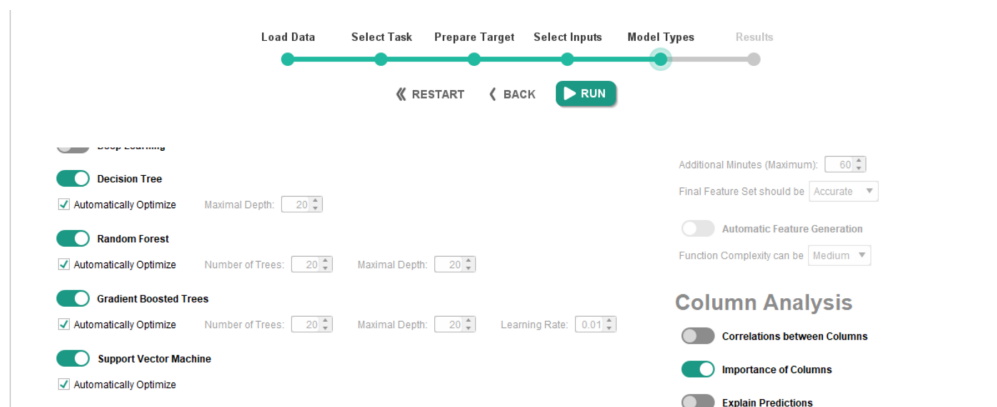


fig 15 Model Selection

## Results

As we can see the results of each model trained and evaluated by Rapid-Miner, we can further export and deploy the models using rapid miner tools.

