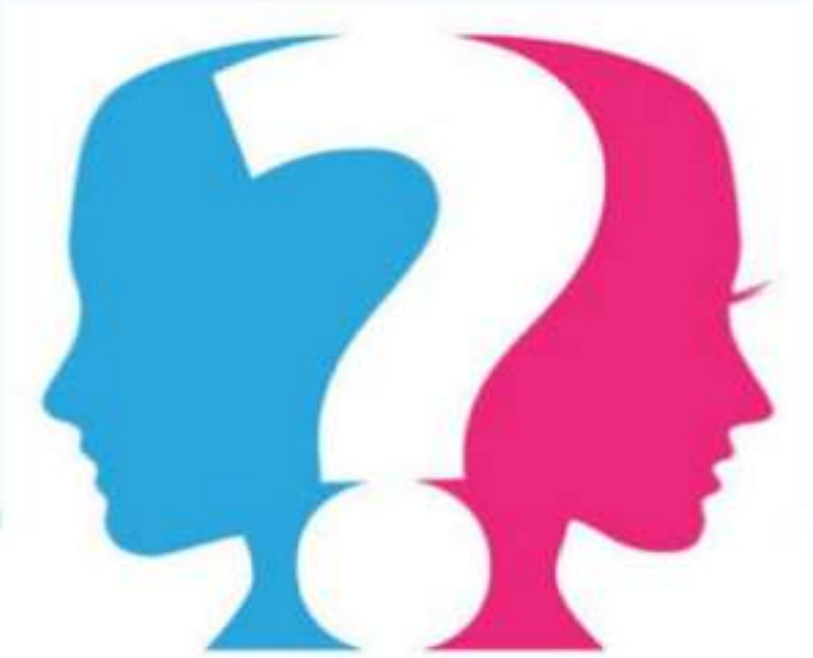
מכון טכנולוגי חולון
Holon Institute of Technology

פרויקט מעשי לתואר B.Sc. במדעים

מתכנתים ממאדים ומהנדסות מנגה

Male programmers from Mars and female engineers from Venus



מגישות: נעם בסט 308465434 ושירן מימון 318274099

בהנחיית: ד"ר יהונתן שלר

תאריך הגשה: 28.07.2021

Table Of Content

Introduction

Studies have shown that there is a difference between male and female in a wide range of fields: biological, physical, mental and intellectual.

Past researches have shown that there is a difference in the manner of expression and writing between the genders.

This project examines wether there are similar differences in writing code between men and women.

The main goal is to answer our research question – which is – Can we predict the programmer's gender from a given code?

The project's data was collected from the *Coderbyte* website. The data was processed and analyzed in advance, features were added to it according to the programming language and modeled using machine learning algorithms.

Project tools

Integrated development environment – We used Jupyter notebook, by Anaconda IDE as our main environment. "The Jupyter Notebook application allows you to create and edit documents that display the input and output of a Python or R language script. Once saved, you can share these files with others." [1]

Programming Language – Python. The python programming language provides a lot of ready to use libraries and methods. "Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects." [2]

Data source web – For our mission, we chose to focus on the Coderbyte website, which provides free challenges and users solutions.

"Coderbyte is **an online collection of 300+ algorithm and full-stack coding challenges and interview kits**. ... They take an

assessment to be rated on their skills in programming language abilities, data structures, and algorithms. Users can move through the ranks by solving challenges correctly. " [3]

CoderByte link - https://coderbyte.com/

Python Libraries

Pandas – The 'DataFrame' collection in Pandas, makes it easier to work with big data and organize it. "Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. " [4]

NumPy – "NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more." [5]

Seaborn – "Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them." [6]

Matplotlib - "Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications.

A Python matplotlib script is structured so that a few lines of code are all that is required in most instances to generate a visual data plot." [8]

BeautifulSoup – We used BeautifulSoup in order to web-crawl in order to obtain our codes. "Beautiful Soup is a Python library for getting data out of HTML, XML, and other markup languages. Say you've found some webpages that display data relevant to your

research, such as date or address information, but that do not provide any way of downloading the data directly. Beautiful Soup helps you pull particular content from a webpage, remove the HTML markup, and save the information. It is a tool for web scraping that helps you clean up and parse the documents you have pulled down from the web." [10]

Selenium – In order to select languages and free codes from the site, we used the selenium library. "Selenium is a free (open-source) automated testing framework used to validate web applications across different browsers and platforms. You can use multiple programming languages like Java, C#, Python etc to create Selenium Test Scripts. Testing done using the Selenium testing tool is usually referred to as Selenium Testing." [9]

OpenCV – In order to compare the user's image with the available emoji's, we used the openCV library. "OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code." [7]

Gender Guesser – In order to detect the user's gender, we used the gender- guesser library, which can detect the gender from a name string. "The program Gender Guesser is a program for determining the gender of a given first name. The program "gender.c" uses the dictionary file "nam_dict.txt" as a data source. This file contains a list of more than 40,000 first names and gender, plus some 600 pairs of "equivalent" names. This list should be able to cover the vast majority of first names in all European contries and in some overseas countries (e.g. China, India, Japan, U.S.A.) as well." [11]

Scikit-learn – We used Scikit-learn in order to use machine learning algorithms, compare the models and predict new examples. "Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy." [12]

Re – When it came to formatting and text-manipulation, we used re in order to extract the specific regular expression we wanted to check in each syntax. "A regular expression is a special sequence of characters

that helps you match or find other strings or sets of strings, using a specialized syntax held in a pattern. Regular expressions are widely used in UNIX world. The Python module re provides full support for Perl-like regular expressions in Python. The re module raises the exception re.error if an error occurs while compiling or using a regular expression." [13]

Scipy – We used SciPy for t testing. "SciPy in Python is an open-source library used for solving mathematical, scientific, engineering, and technical problems. It allows users to manipulate the data and visualize the data using a wide range of high-level Python commands. SciPy is built on the Python NumPy extention." [14]

Project planning and structure

Generic Flow

Extract Codes

User_url	Challange_Name	Programming_Lang	Sol_Code	User_Score	User's_Comments
/profile/Salapet	Bracket Combinations	Java	import java.util.*; \nimport java.io.*;\n\ncla...	15	5
/profile/nvegesna	Bracket Combinations	Java	import java.util.*; \nimport java.io.*;\n\ncla...	10	1
/profile/omegatau	Bracket Combinations	Java	import java.util.*;\nimport java.io.*;\n\nclas...	15	0
/profile/dastapati	Bracket Combinations	Java	import java.util.*; \nimport java.io.*;\n\ncla...	15	0
/profile/neo360	Bracket Combinations	Java	class BracketCombinations:\n\n def __init__(s...	15	0

Clean and format data + Feature enrichment

Name_or_Image	...	While	Func dec	Varibales Dec	Comments length	code length
1	...	1	2	2	6	120
0	...	0	4	1	6	96
0	...	0	4	0	6	66
1	...	0	3	2	42	91
0	...	0	3	2	10	60

Train the models and predict the gender

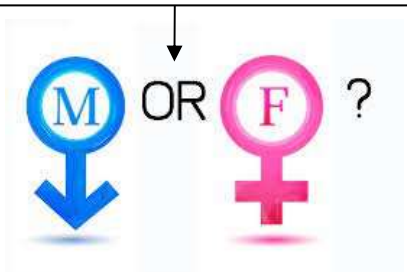


Figure 1 Generic System's Flow

This project was divided into three main sections:

1. **Section 1: -** Web_Crawling_Gender_Detection.ipynb

   In this section, we focused on obtaining the data and detecting the user's gender, in order to label and classify each record.

   https://github.com/noambassat/Male-programmers-from-Mars-and-female-engineers-from-Venus/blob/main/Web_Crawling_Gender_Detection.ipynb

2. **Section 2** - Syntaxes.ipynb:

   This section is mainly about data preprocessing and enrichment: cleaning the raw data, dropping duplicate records, reformatting each column, and normalizing units. In addition, we added many features for each record that represent the code's details.

   a. Langs_Syntax
   b. CountVectorize

   https://github.com/noambassat/Male-programmers-from-Mars-and-female-engineers-from-Venus/blob/main/Syntaxes.ipynb

3. **Section 3 -** Modeling Data.ipynb:

   Training different Machine Learning models, comparing and testing.

   a. Grid search
   b. Cross validation
   c. Comparing by accuracies and f1_score
      i. DT
      ii. RF
      iii. KNN
      iv. SVM
      v. NB

   https://github.com/noambassat/Male-programmers-from-Mars-and-female-engineers-from-Venus/blob/main/Modeling%20Data.ipynb

Web Crawling flow:

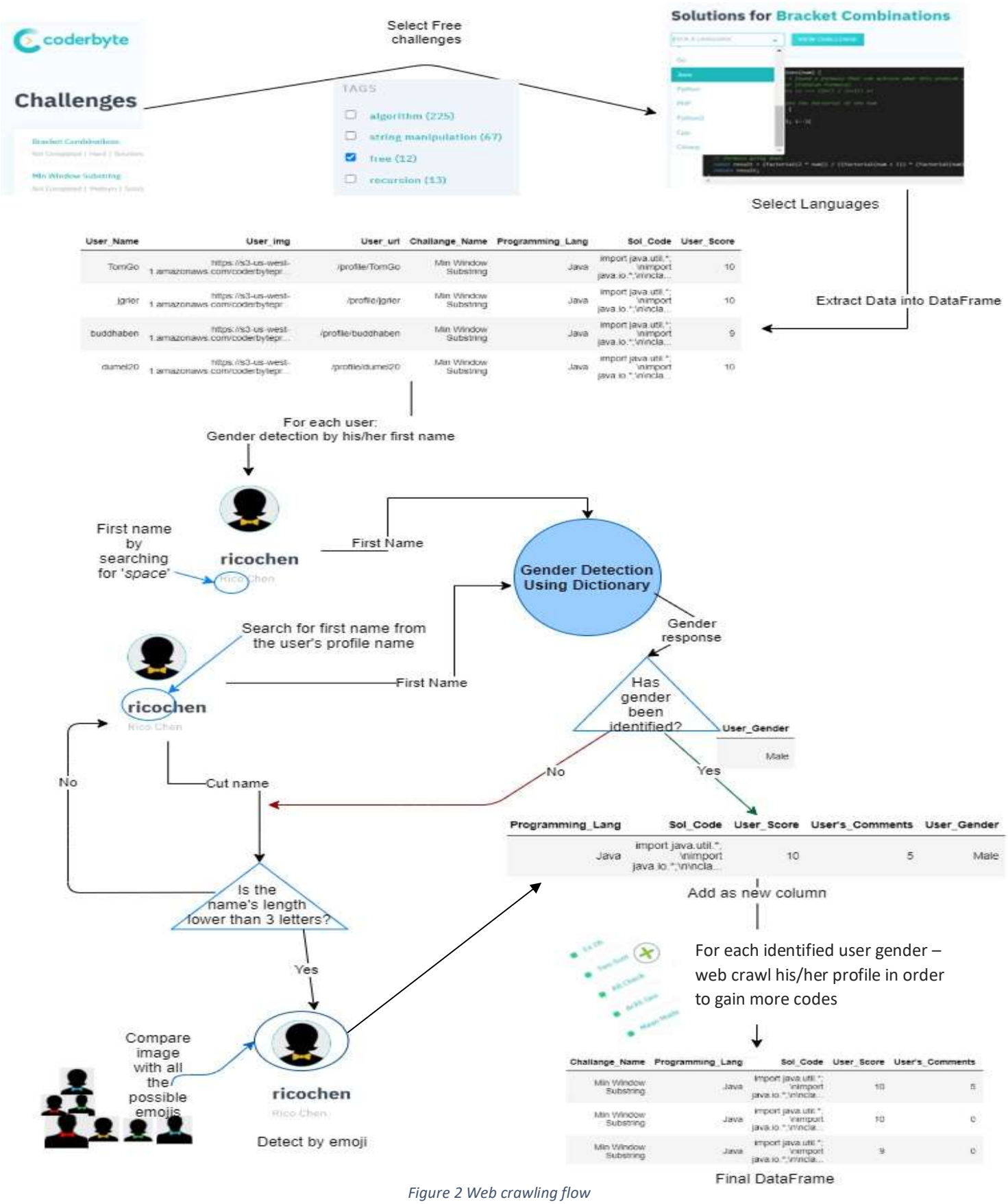


Figure 2 Web crawling flow

Web Crawling Description:

Obtaining the codes from the Coder-Byte website, detecting the writer's gender, and re-crawling each user's profile in order to gain more code solutions.

a. <u>Crawl free challenges solutions:</u>

   Our main tool for web crawling was "BeautifulSoup" library, but in order to select only the free challenges and select specific code languages, we also used the Selenium library.

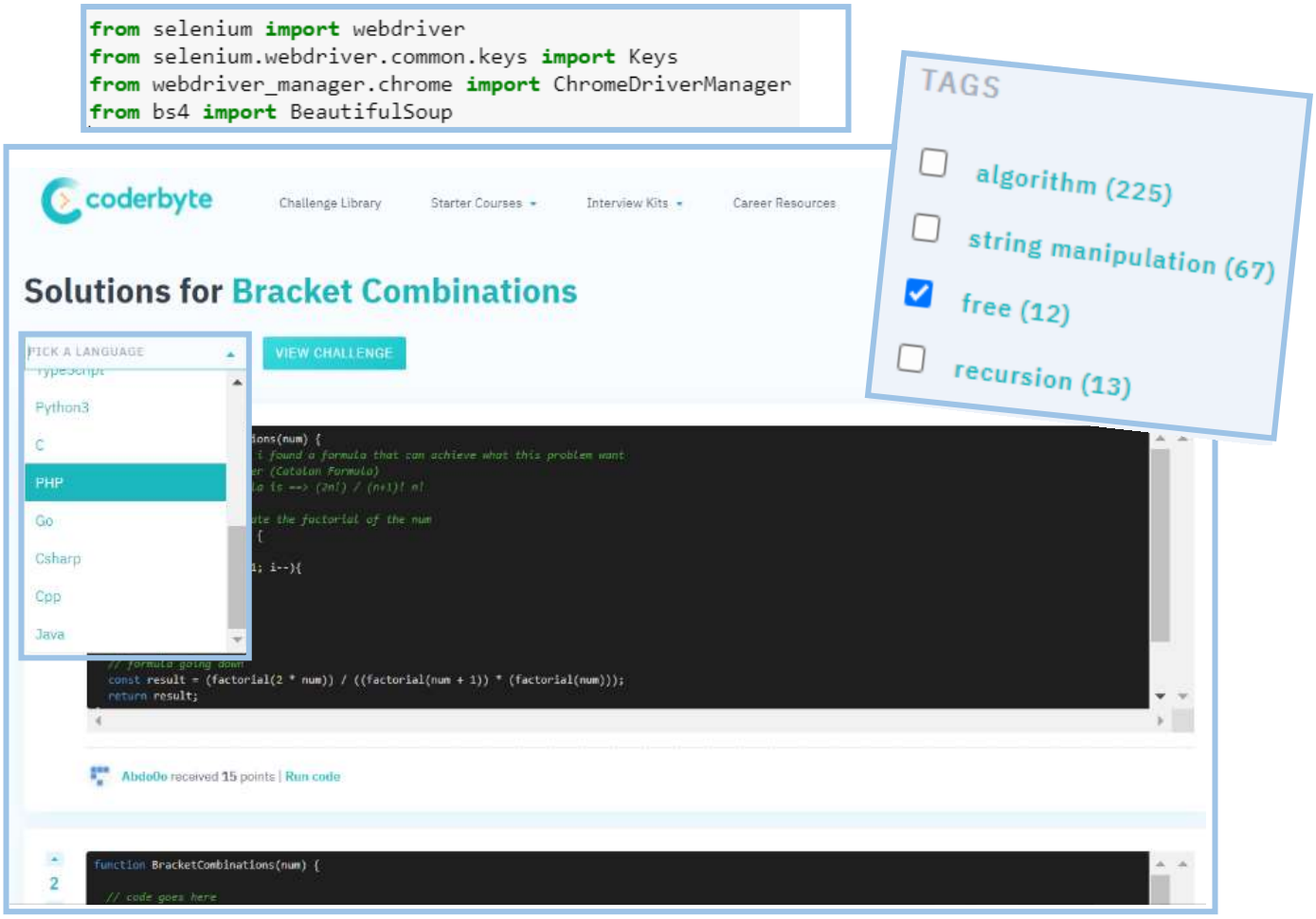


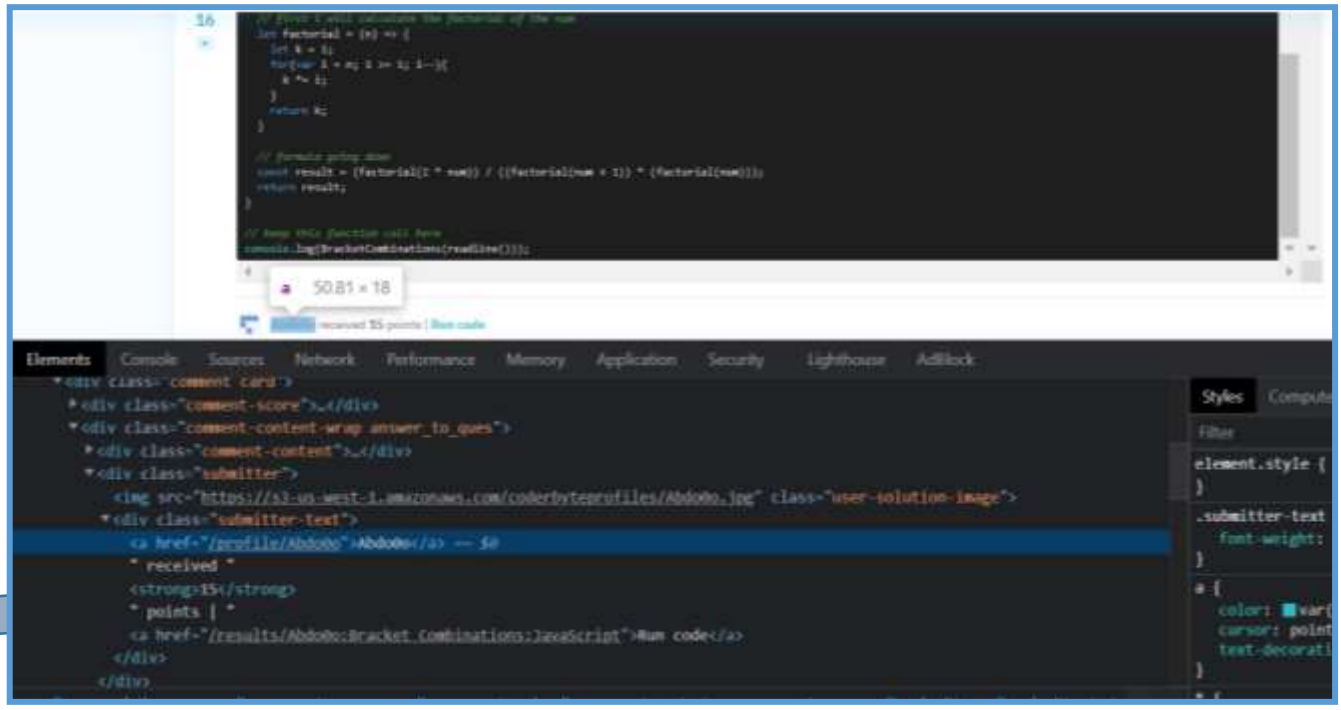
Figure 3 BeautifulSoup & Selenium

Figure 4. The web-crawling process

We chose to focus on nine main programming languages - *Java, Python, JavaScript, PHP, Csharp, Swift, C, Ruby and Go.*

We saved all the data in a data-frame structure, which initially contained the following columns:

User Name, User image, User profile URL link, Challenge Name, Programming Language, Solution Code, User Score and User's Comments.



User_Name	User_img	User_url	Challange_Name	Programming_Lang	Sol_Code	User_Score	User's_Comments
Salapet	https://s3-us-west-1.amazonaws.com/coderbytepr...	/profile/Salapet	Bracket Combinations	Java	import java.util.*; \nimport java.io.*;\n\ncla...	15	5
nvegesna	https://s3-us-west-1.amazonaws.com/coderbytepr...	/profile/nvegesna	Bracket Combinations	Java	import java.util.*; \nimport java.io.*;\n\ncla...	10	1
omegatau	https://s3-us-west-1.amazonaws.com/coderbytepr...	/profile/omegatau	Bracket Combinations	Java	import java.util.*;\nimport java.io.*;\n\nclas...	15	0
dastapati	https://s3-us-west-1.amazonaws.com/coderbytepr...	/profile/dastapati	Bracket Combinations	Java	import java.util.*; \nimport java.io.*;\n\ncla...	15	0
neo360	https://s3-us-west-1.amazonaws.com/coderbytepr...	/profile/neo360	Bracket Combinations	Java	class BracketCombinations:\n\n def __init__(s...	15	0
...	...	...	...	...	...	...	...
husharma18	https://s3-us-west-1.amazonaws.com/coderbytepr...	/profile/himanshusharma18	First Reverse	Java	import java.util.*; \nimport java.io.*;\n\ncla...	10	0
jrorsini	https://s3-us-west-1.amazonaws.com/coderbytepr...	/profile/jrorsini	First Reverse	Java	def FirstReverse(str): \n\n # code goes her...	10	0
bneeriemer	https://s3-us-west-1.amazonaws.com/coderbytepr...	/profile/bneeriemer	First Reverse	Java	import java.util.*; \nimport java.io.*;\n\ncla...	10	0
eviousBard	https://s3-us-west-1.amazonaws.com/coderbytepr	/profile/DeviousBard	First Reverse	Java	import java.util.*; \nimport java.io.*;\n\ncla	10	0

Figure 5 Data Extraction

b. Gender Detection:

Sequence detection steps:

- *Detect by first name*

  At first, the program checks if there is a name in the user's profile, **if** there is – try to detect the first name (first string until the 'space' char or non-alpha char) by the dictionary detector. **Else**, continue to the next step.

Figure 6. First Name Examples

- *Detect by username*

  If the first name was missing, or the detector could not recognize the gender, try to detect the gender by username, in the following sequence:

  a. Slice the name by capital letter (As long as the capital letter is not the first letter).

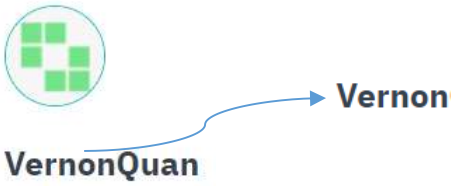


Figure 7. Username Example

b. Slice name by non-alpha character

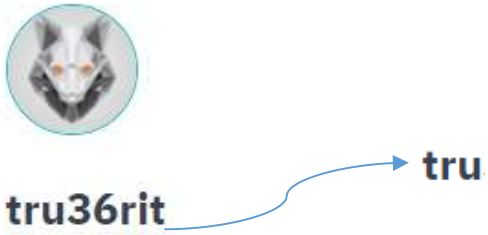


Figure 8. Slice by non-alpha example

c. Slice last letter each time. Continue as long as the gender is not recognized and the username's length is greater than three letters.

Figure 9. Slice last letter

Detect by emoji image

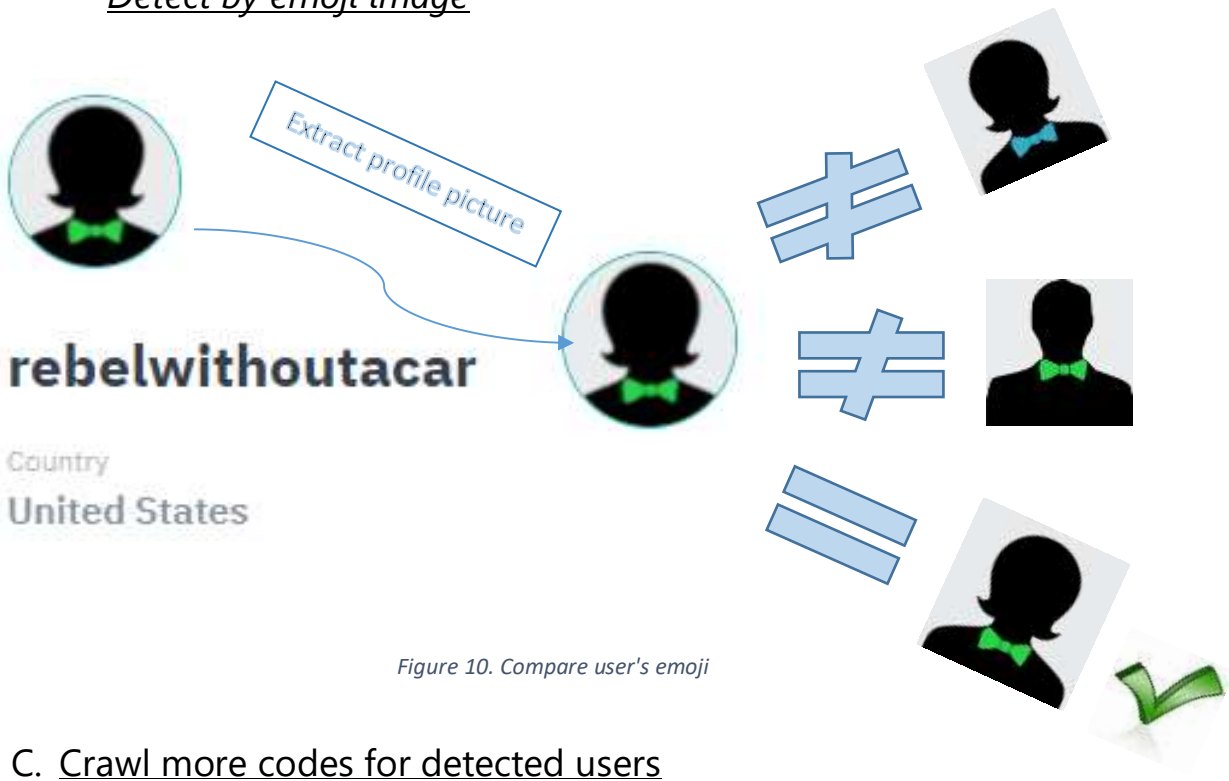


Figure 10. Compare user's emoji

C. Crawl more codes for detected users

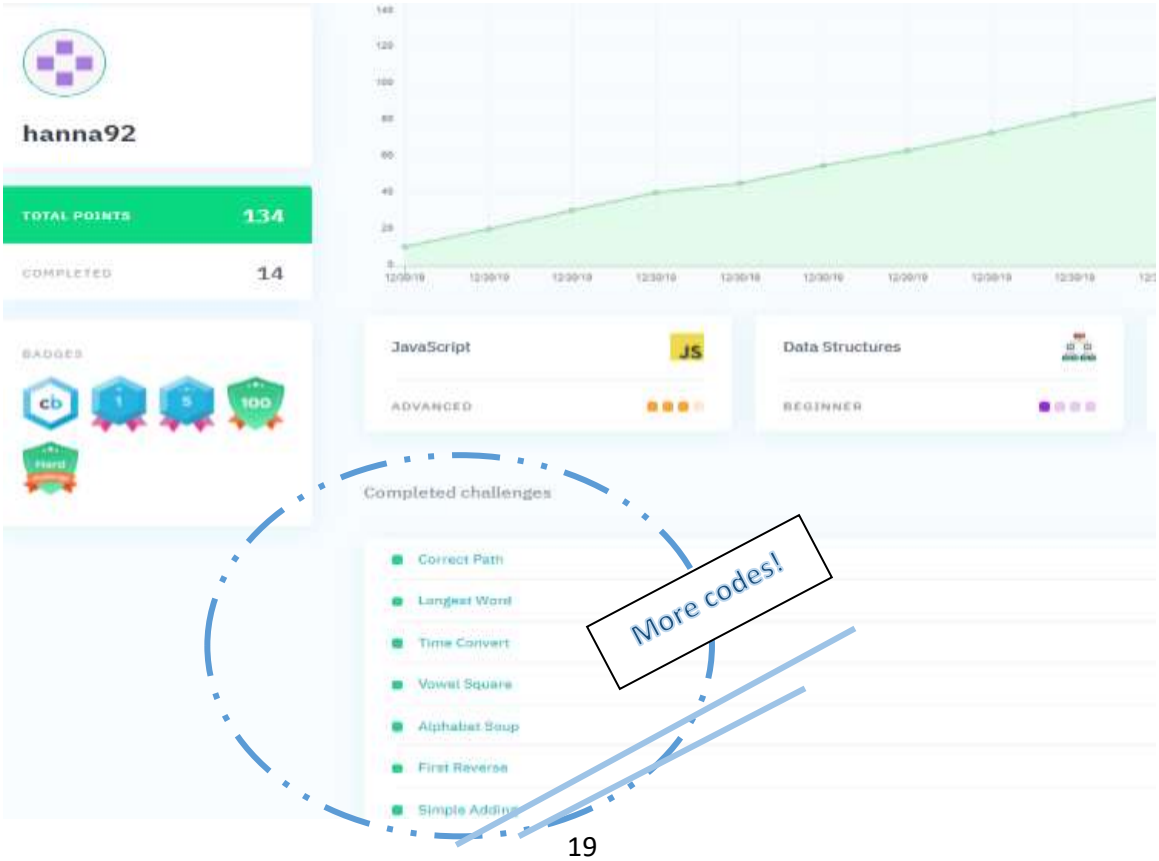


Figure 11. Gain more codes

Initial Features:

- User_Name

- User_img

- User_url – user's profile link
- Challange_Name

- Programming_Lang

- Sol_Code

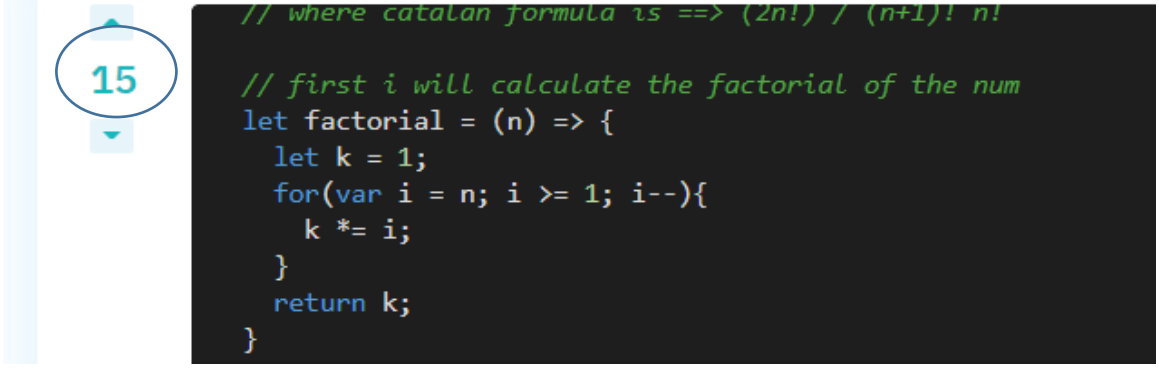
```
function BracketCombinations(num) {
  let count = 0;
  (function inner(left, right, str) {
      if (left === 0 && right === 0) {
      count++;
    }

    if (left > 0) {
      inner(left - 1, right + 1, str + '(');
    }

    if (right > 0) {
      inner(left, right - 1, str + ')');
    }
  })(num, 0, '');
  return count;
}
BracketCombinations(readline());
```

- User_Score

- User's_Comments



- User_Gender – After gender detection:

Male or Female

- Name_or_Image - whether the gender was detected by the user's name or image:

  0 Means detected by name, 1 means by image.

Percentage of genders

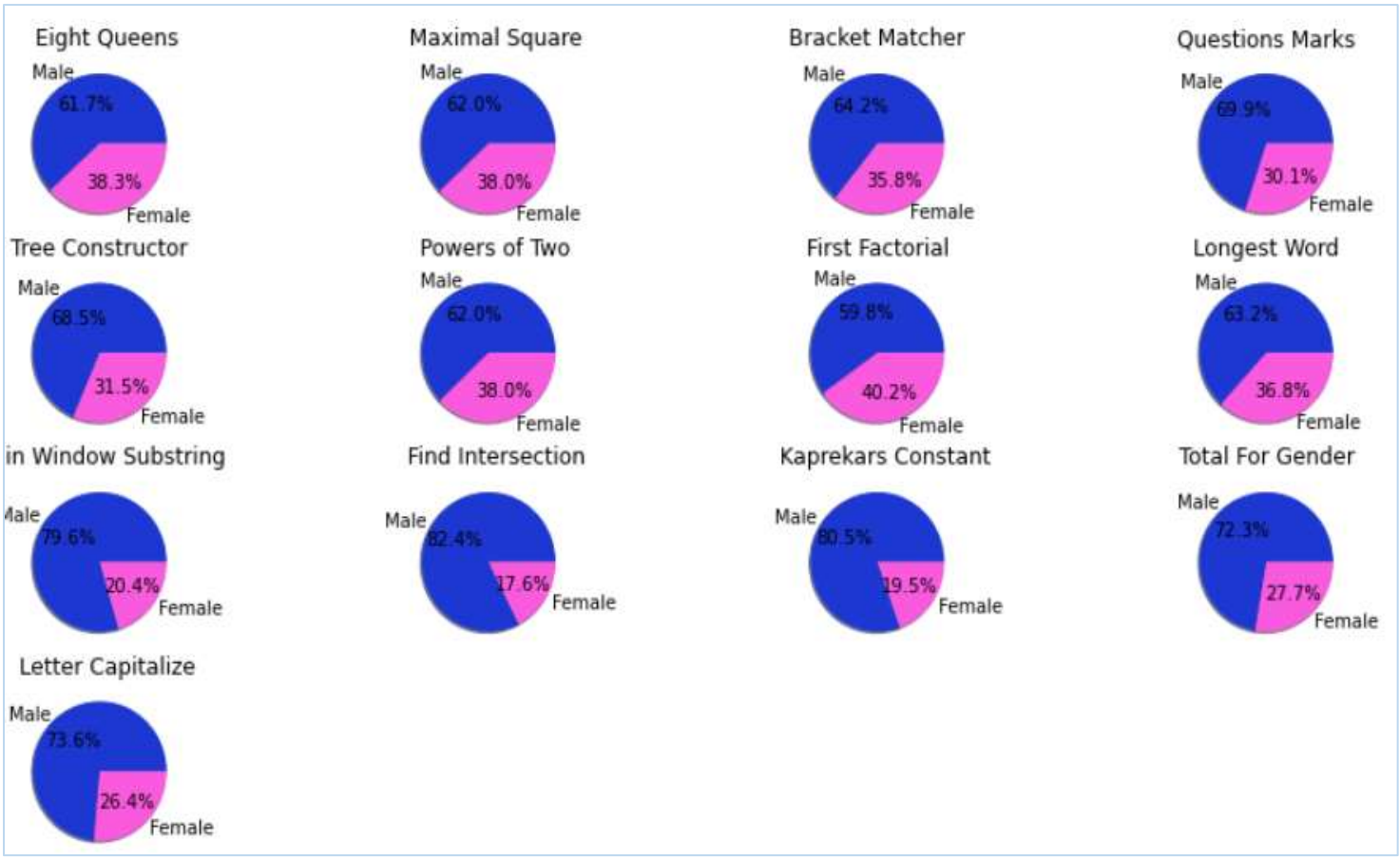


Figure 12. Pie chart: Percentage of genders in each challenge

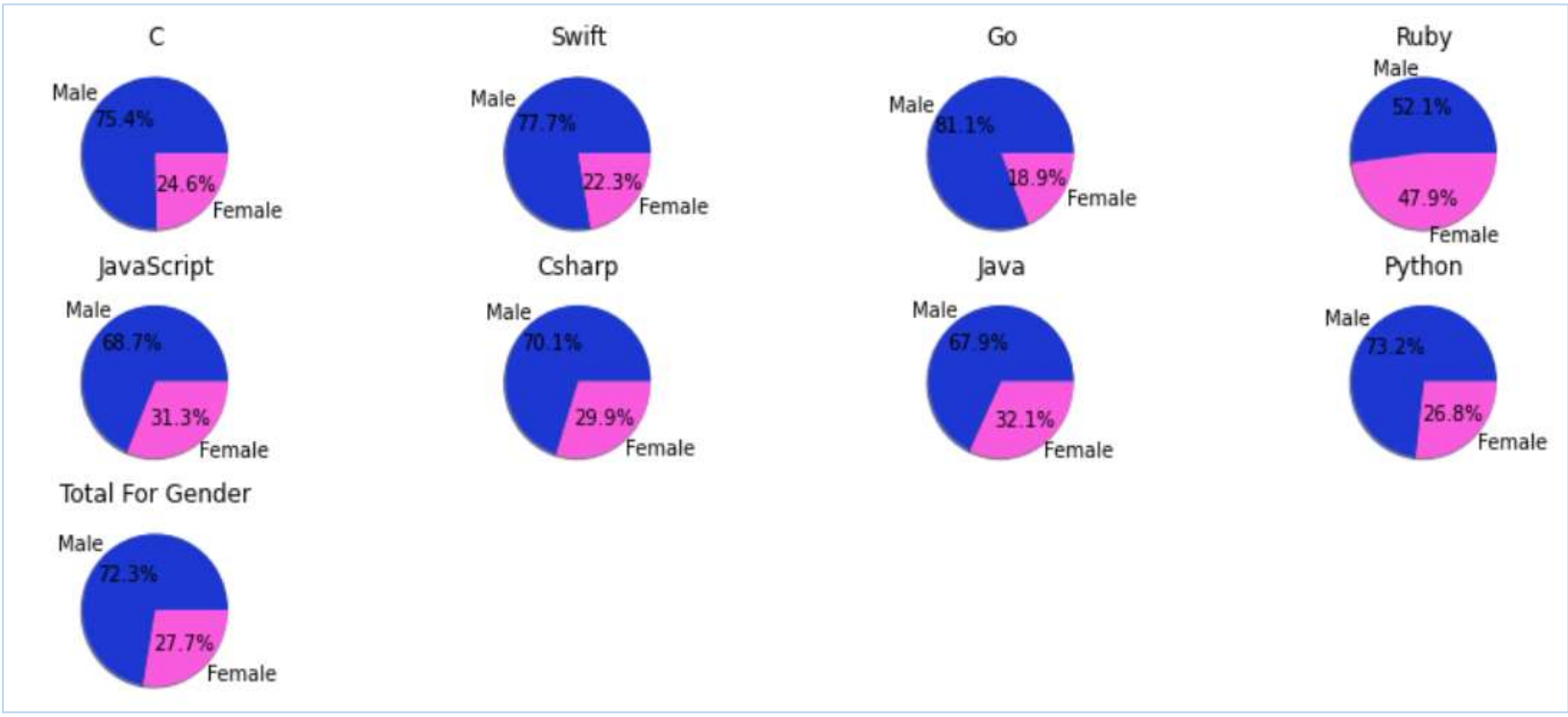


Figure 13. Pie chart: Percentage of genders in each programming language

Syntax Features flow:

	Command	C	Python	Swift	PHP	Ruby	Go	JavaScript	Csharp	Python3	Java
0	Comments	// /*	# """	// /*	// # /*	# =begin	// /*	// /*	// /*	# """	// /*
1	Conditions	if if(case ? else else(else	if else else: elif	if else	if if(else	if unless else elsif elsif	if if(case else else(else	if if(case else else(else	if if(case ? else else(else	if else else: elif	if if(case ? else else(else
2	For	for for(	for	for	for for(foreach(foreach	for	for	for for(	for for(	for	for for(
3	While	while while(	while while(	while	while while(	while	NaN	while while(	while while(	while while(	while while(
5	Func dec	return void	def	func	function	def	func	function function(	return void	def	return void
6	Varibales Dec	double int char String string float boolean lo...	<var_name>= <val_name>	var let	$ const	$ @ @@ _	var	let var	double int char String float boolean long shor...	<var_name>= <var_name> =	double int char String float boolean long shor...

Create Syntax DataFrame for each language

Create New Features

Add new columns to the original DF

While	Func dec	Varibales Dec	Comments length	code length
1	2	2	6	120
0	4	1	6	96
0	4	0	6	66

Extract all the variables name and count their in each code using Count-Vectorizer, add as new features to the original DF

words	x	y
0	0	0
0	0	0
0	0	0
0	0	0

User_Score	User's_Comments	User_Gender	Name_or_Image	...	value	we	wo	wor	word	word	words
15	0	0	0	...	0	0	0	0	0	0	0
9	0	1	1	...	0	0	0	0	0	0	0
15	0	0	0	...	0	0	0	0	0	0	0
15	0	0	0	...	0	0	0	0	0	0	0
14	0	0	0	...	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...
10	0	0	0	...	0	0	0	0	0	0	0
10	0	0	0	...	0	0	1	1	1	0	1
10	0	0	0	...	0	0	0	0	0	0	0

Final DataFrame

Code & Language

Clean Data

Data processing

Drop none, Drop duplicates codes, normalizing units, reformatting strings into numeric values

Raw Data

User_Name	User_img	User_url	Challange_Name	Programming_Lang	Sol_Code	User_Score
TomGo	https://s3-us-west-1.amazonaws.com/codebytepr...	/profile/TomGo	Min Window Substring	Java	import java.util.*; import java.io.*;\n\nclass...	10
gnee	https://s3-us-west-1.amazonaws.com/codebytepr...	/profile/gnee	Min Window Substring	Java	import java.util.*; import java.io.*;\n\nclass...	10
buddhabon	https://s3-us-west-1.amazonaws.com/codebytepr...	/profile/buddhabon	Min Window Substring	Java	import java.util.*; import java.io.*;\n\nclass...	9
dumet20	https://s3-us-west-1.amazonaws.com/codebytepr...	/profile/dumet20	Min Window Substring	Java	import java.util.*; import java.io.*;\n\nclass...	10

Original DataFrame

Figure 14. Syntax Features flow

Syntax Features Description:

Commands features:

After conducting a background research on the syntax of the various programming languages, we created a table where the rows represent some basic commands while the columns represent each language. Based on this syntax table, we created additional features:

- Code length
- Number of comments
- The location of comments in code
- The amount of variables in code
- The amount of FOR and WHILE loops used
- Number of functions
- Number of uses in IF and ELSE conditions
- Number of rows
- Number of capital letters in variable names
- The use of permissions (Public, private, protected)
- The use of 'static'
- Number of blank lines
- Number of default comments (whether the user deleted or left them)

- Print / output commands count
- Counting the different ways to write comments (e.g. in Java, you can write / * comment * / or write // comment).

	Command	C	Python	Swift	PHP	Ruby	Go	JavaScript	Csharp	Python3	Java
0	Comments	// /*	# """	// /*	// # /*	# =begin	// /*	// /*	// /*	# """	// /*
1	Conditions	if if(case ? else else{ else	if else else: elif	if else	if if(else	if unless else elseif elsif	if if(case else else{ else	if if(case else else{ else	if if(case ? else else{ else	if else else: elif	if if(case ? else else{ else
2	For	for for(	for	for	for for(oreach(foreach	for	for	for for(	for for(	for	for for(
3	While	while while(	while while(	while	while while(	while	NaN	while while(	while while(	while while(	while while(
5	Func dec	return void	def	func	function	def	func	function function(	return void	def	return void
6	Varibales Dec	double int char String string float boolean lo...	<var_name>= <var_name>	var let	$ const	$ @ @@ _	var	let var	double int char String float boolean long shor...	<var_name>= <var_name> =	double int char String float boolean long shor...

Figure 15. The Syntax Table Data Frame

While	Func dec	Varibales Dec	Comments length	code length	Variables Names	Num_of_Lines	Lines_ave_len	Vars_ave_len	num_of_cases_uses
0	4	1	0	87	'leftBracketsNumber,'	31	0	18	1
1	2	2	6	120	'strVal' 'combination'	48	0	8	1
0	10	0	22	156		46	0	0	0
0	2	5	15	116	'possibleCombinations' 'combination;' 'validC...	45	0	16	4
0	6	0	6	131		42	0	0	0
...	...	...	...	...	...	...	...	...	...
0	2	0	31	31		1	0	0	0

Figure 16. Data Frame after adding features

Variable names features:

In order to count the existence of each variable name, we used CountVectorizer.

"CountVectorizer is a great tool provided by the scikit-learn library in Python. It is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text. This is helpful when we have multiple such texts, and we wish to convert each word in each text into vectors (for using in further text analysis)." [15]

```
from sklearn.feature_extraction.text import CountVectorizer
```

	at	atc	atch	ax	ax	ay	ay	b	be	ber	...	value	we	wo	wor
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0
1	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0	...	0	0	0	0
3	9	0	0	0	0	0	0	0	0	0	...	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0

Figure 17. CountVectorizer Features Example

Normalized vs. Regular DataFrames:

In order to examine the best performance for our mission, we created two types of the main DataFrame:

1. A normalized dataframe – where each value is divided by the specific challenge's code average length.
2. The original dataframe.

Bar Graphs of the differences between the genders in each language.

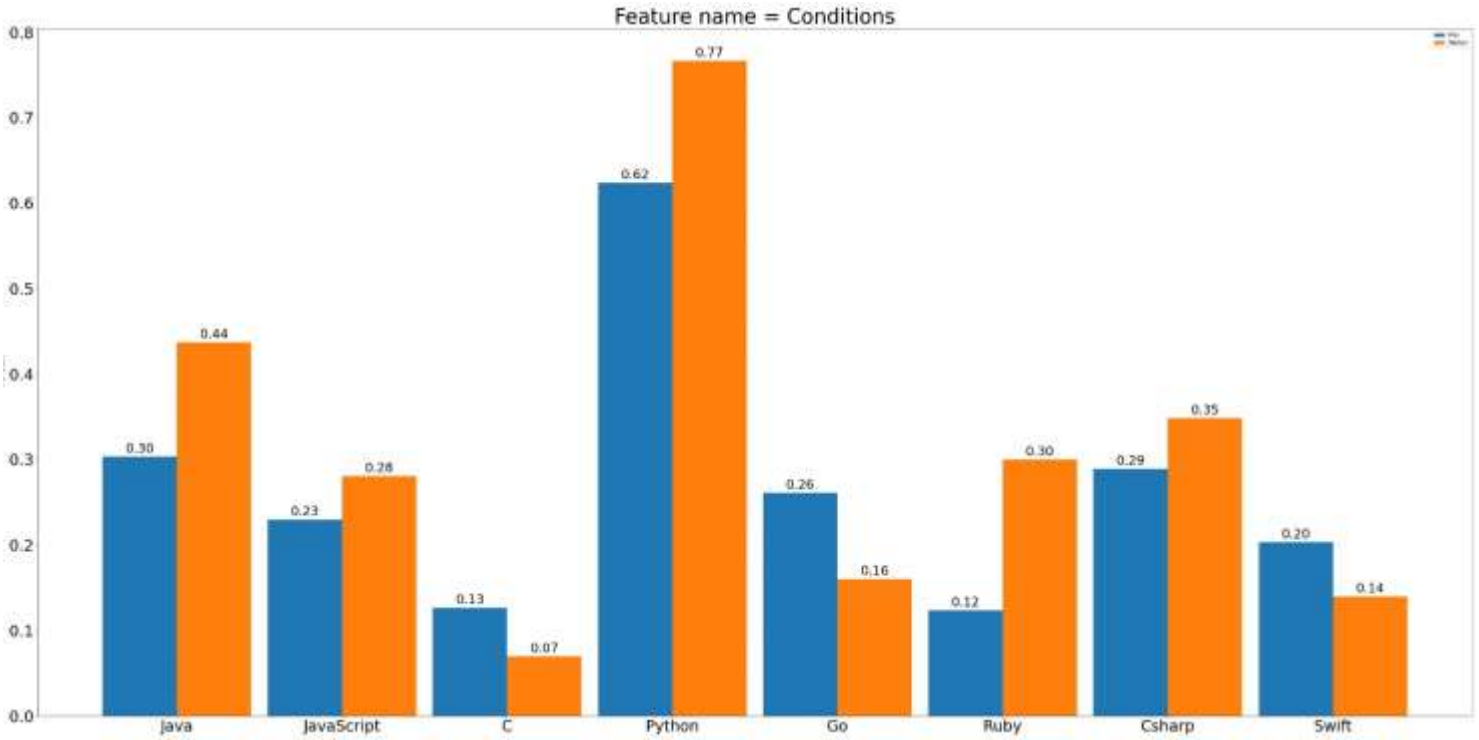


Figure 18. Conditions uses by gender

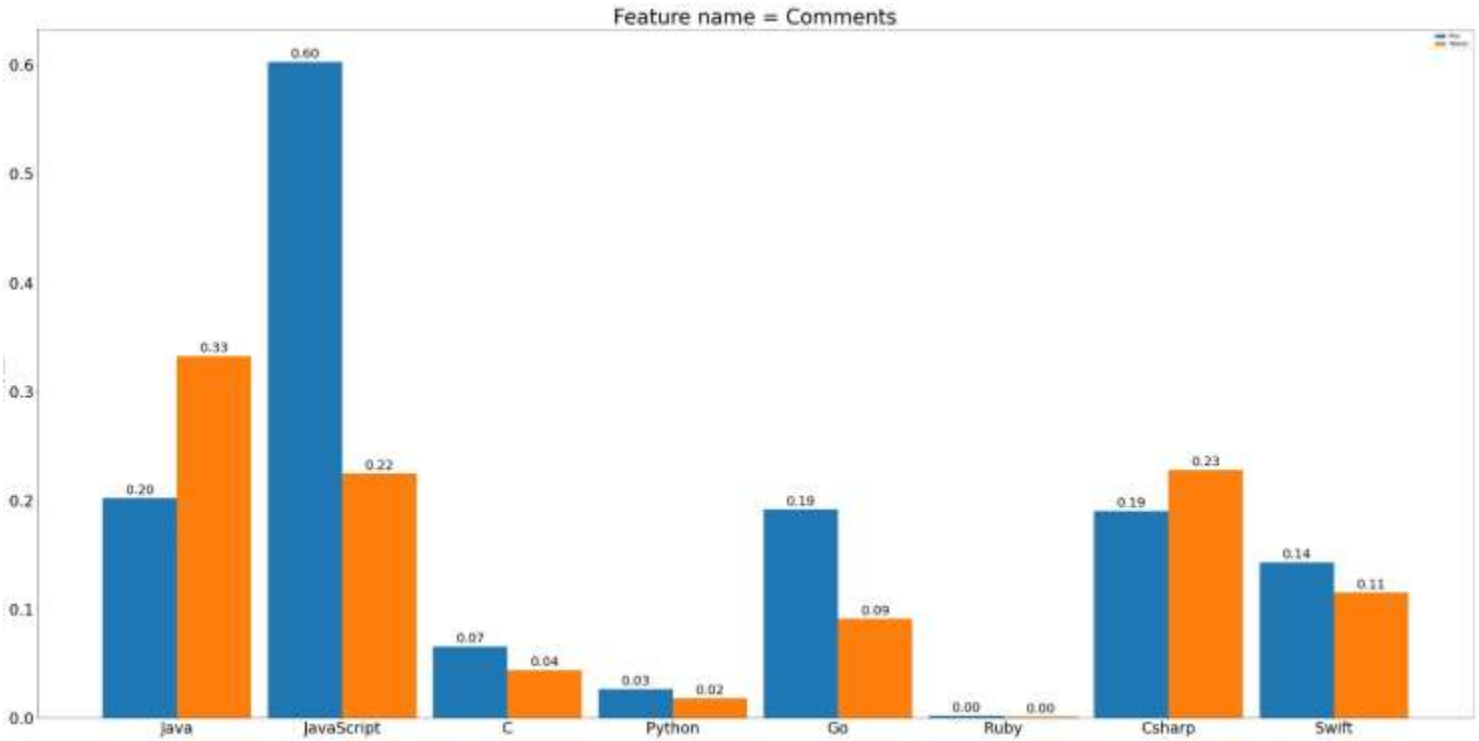


Figure 19.Number of comments by gender

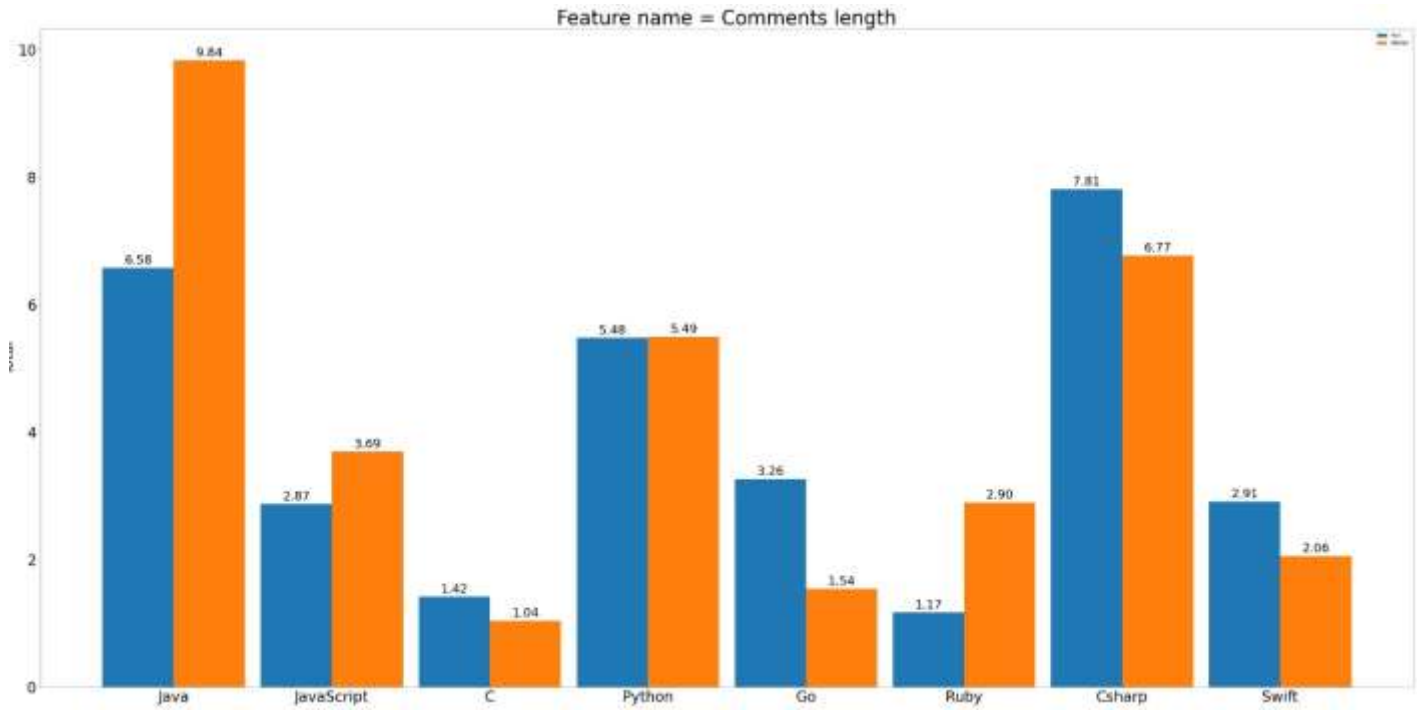


Figure 20. Average comments length by gender

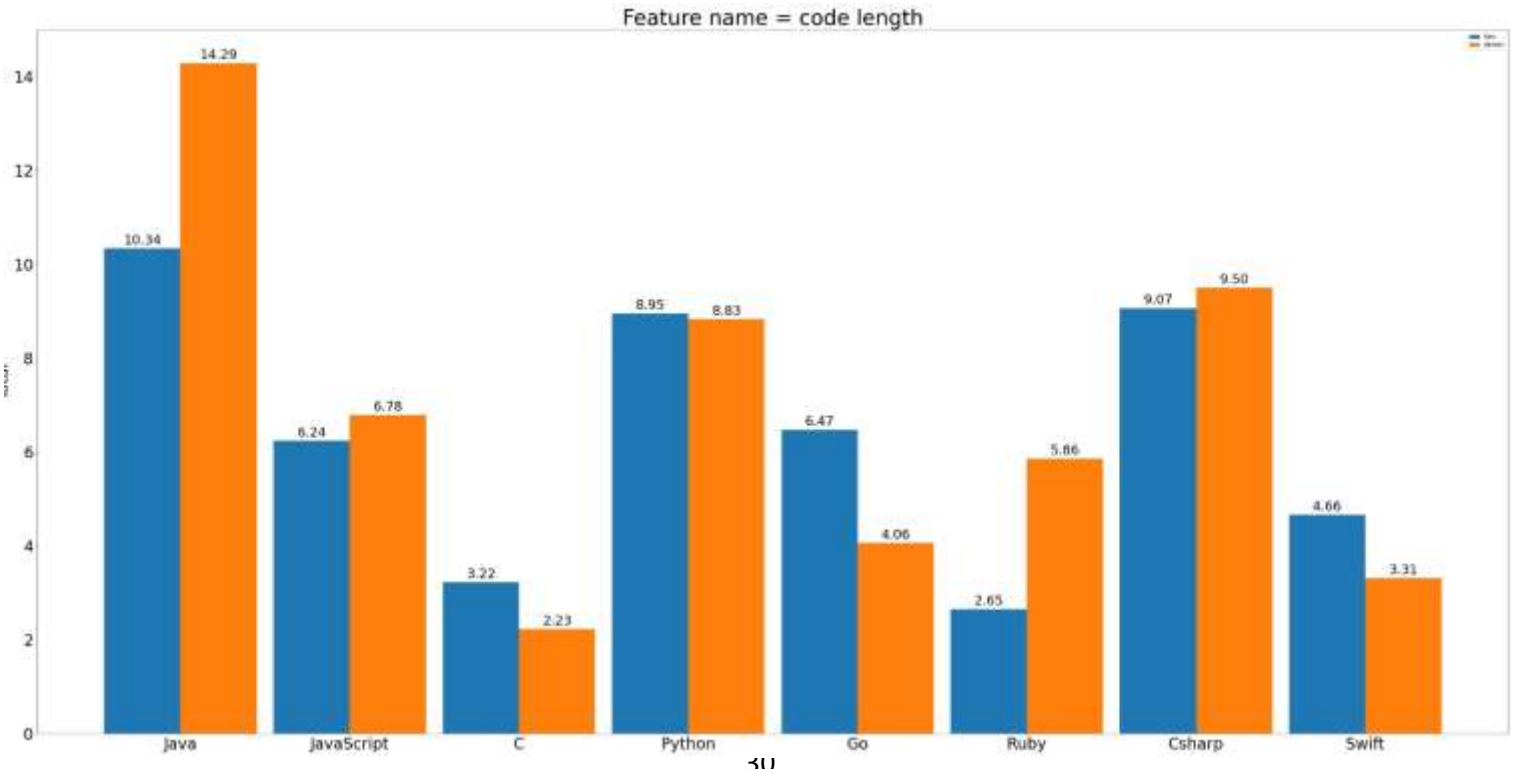


Figure 21. Average Code length by gender

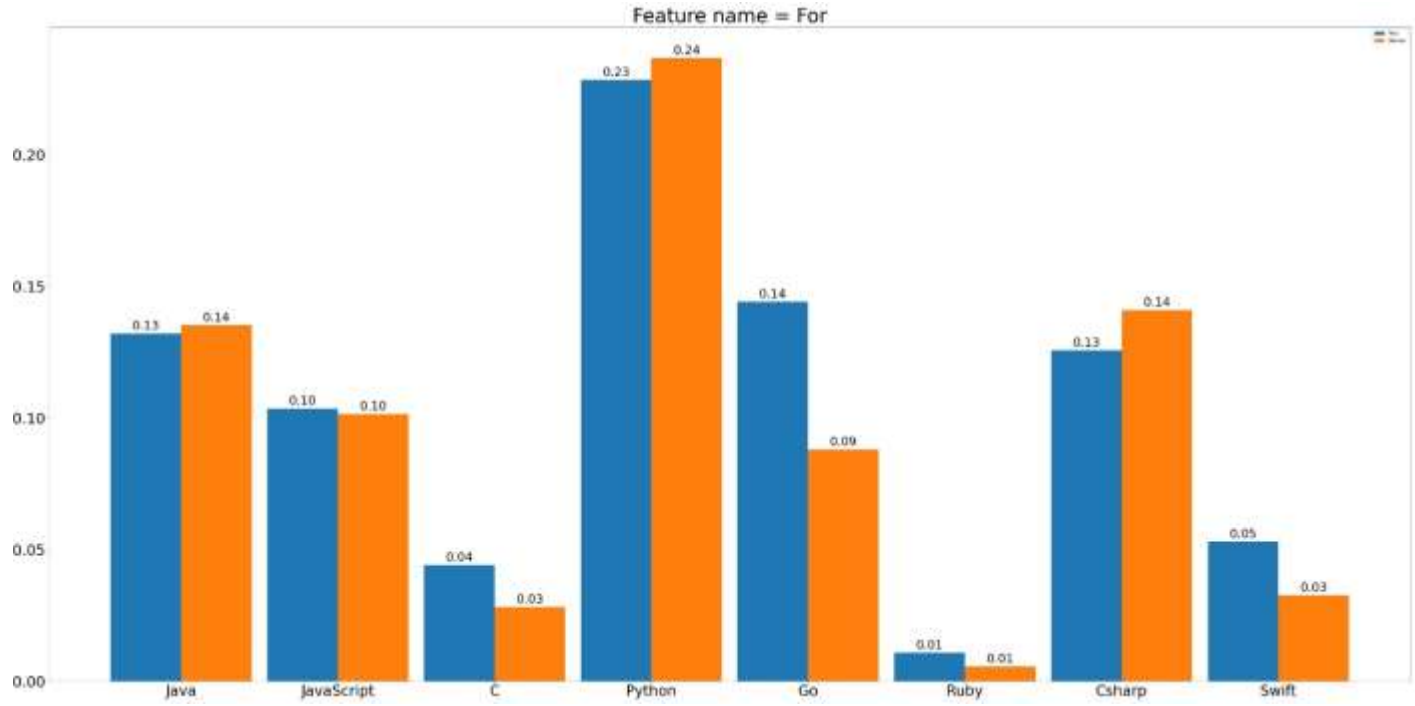


Figure 23. Average for loops uses by gender

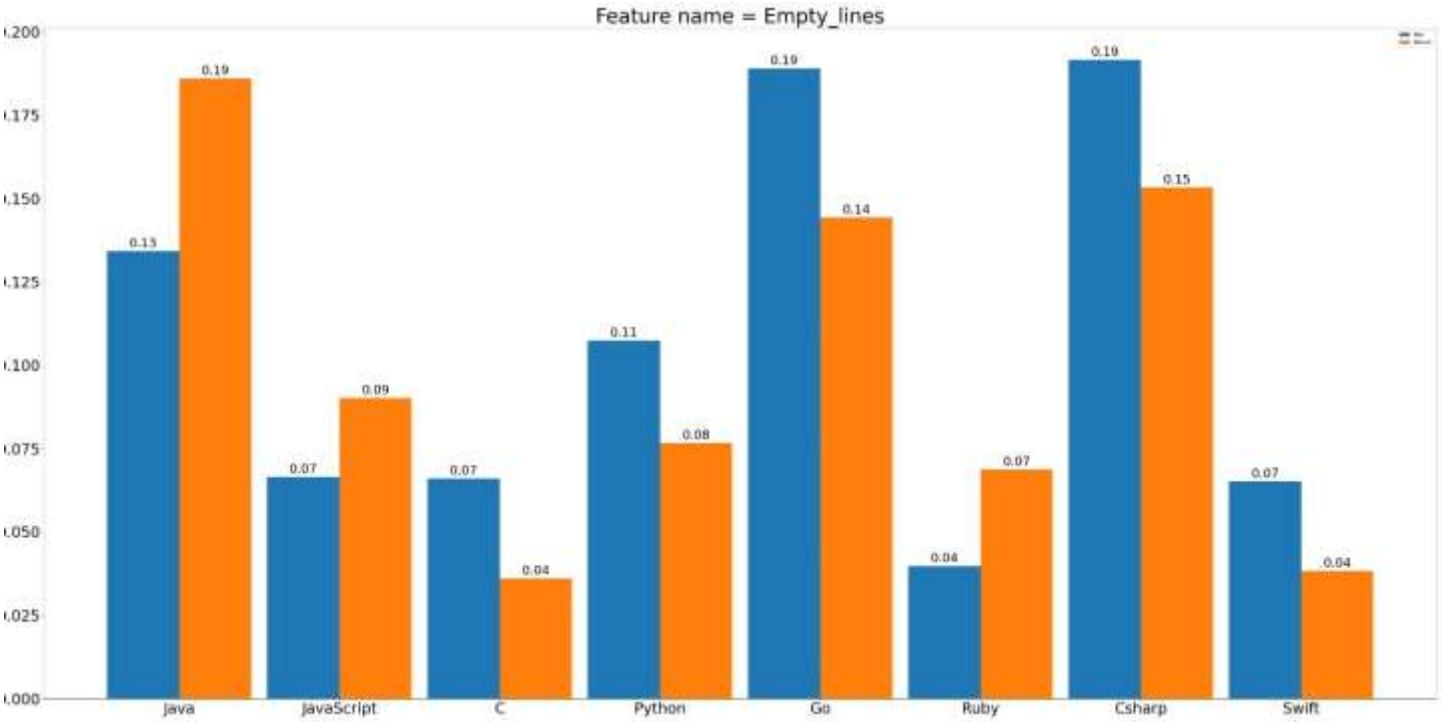


Figure 22. Average number of empty lines by gender

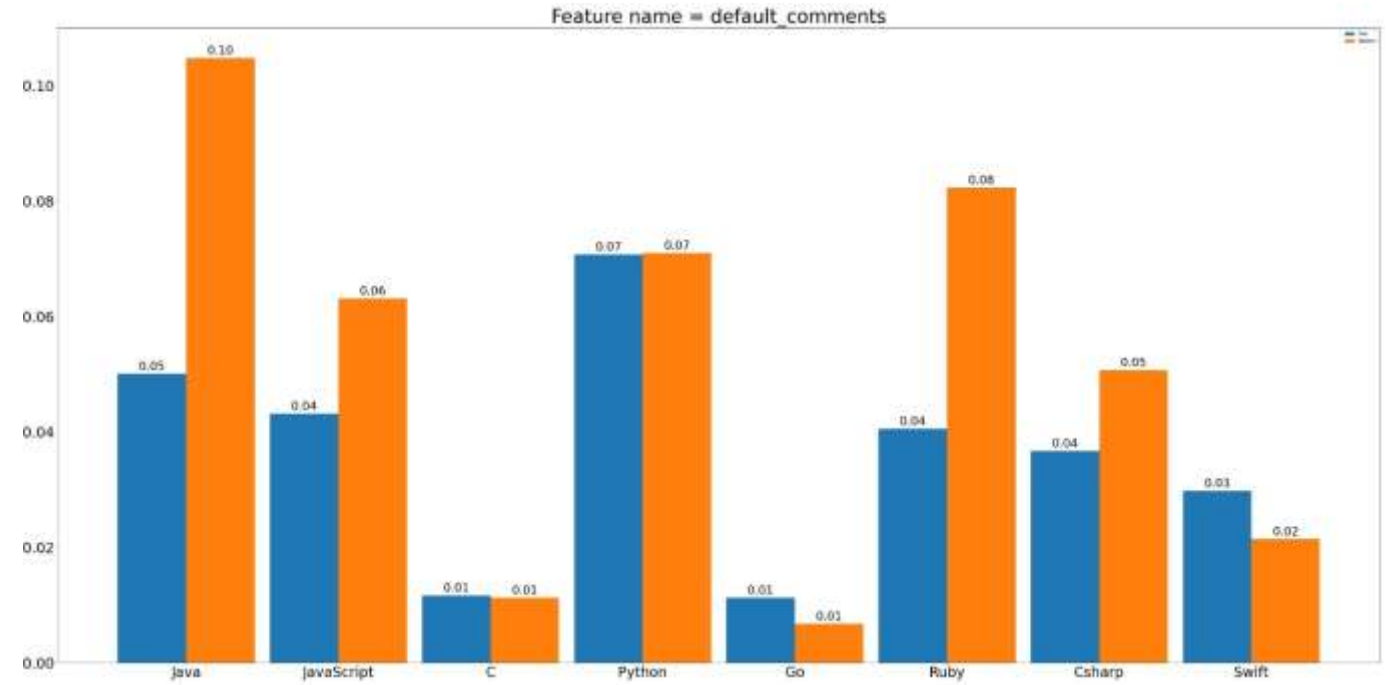


Figure 25. Average number of default comments by gender

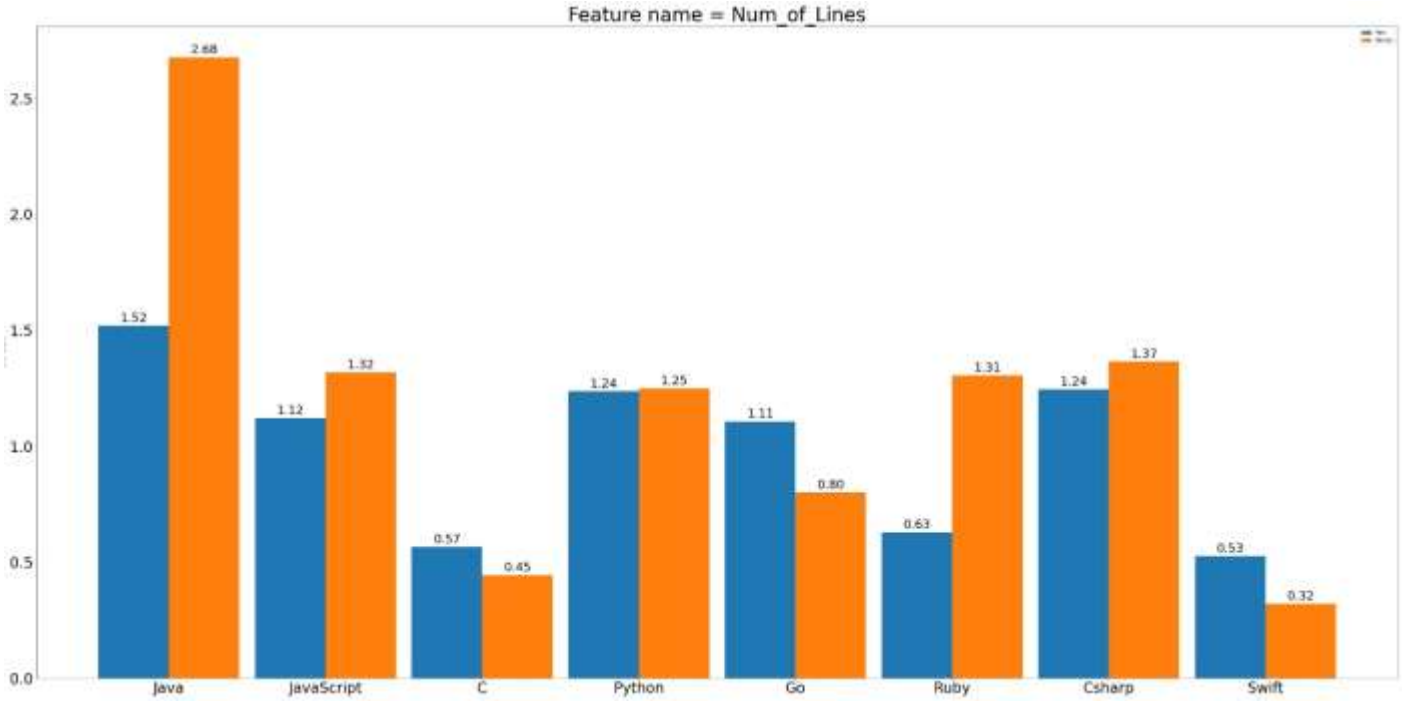


Figure 24. Average number of total lines by gender

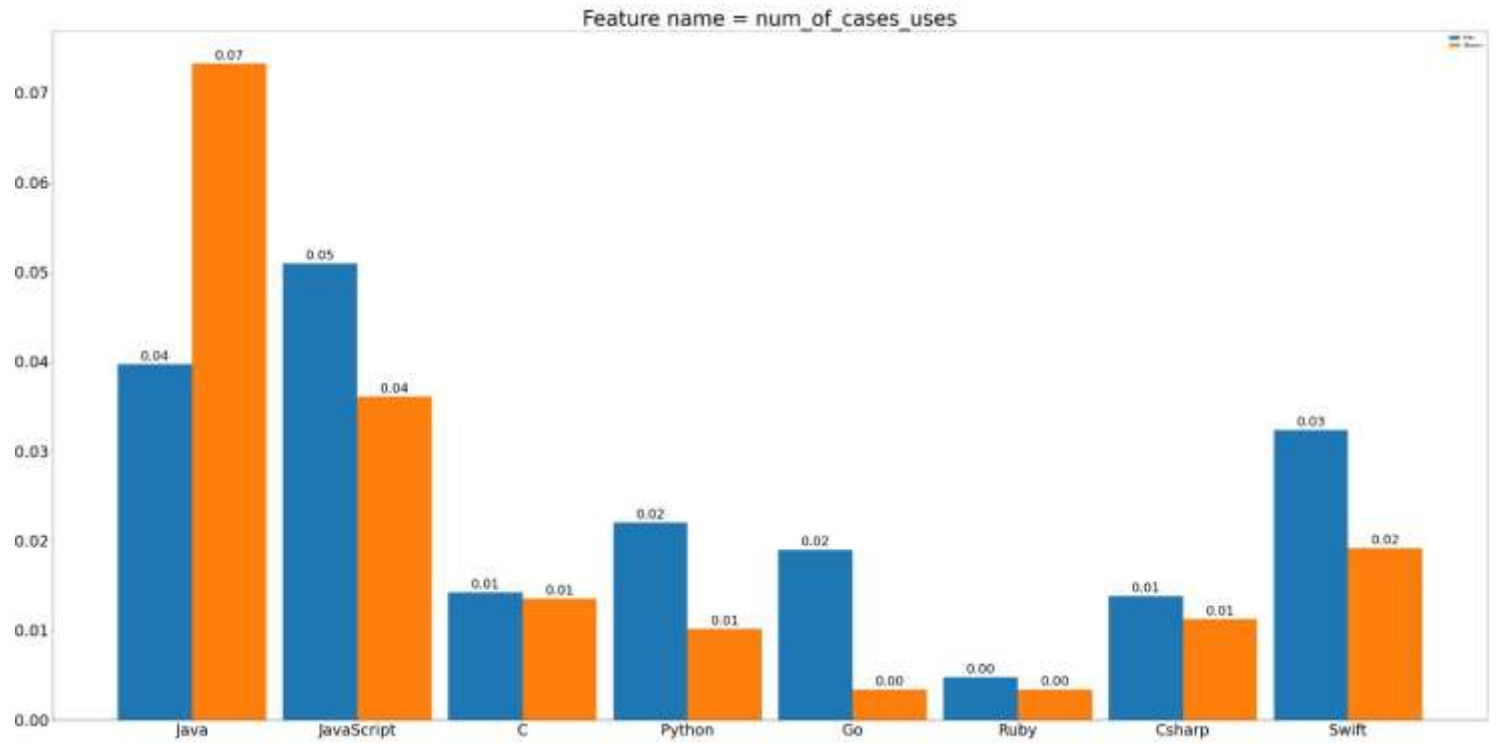


Figure 27. Average number of Capital letters uses by gender

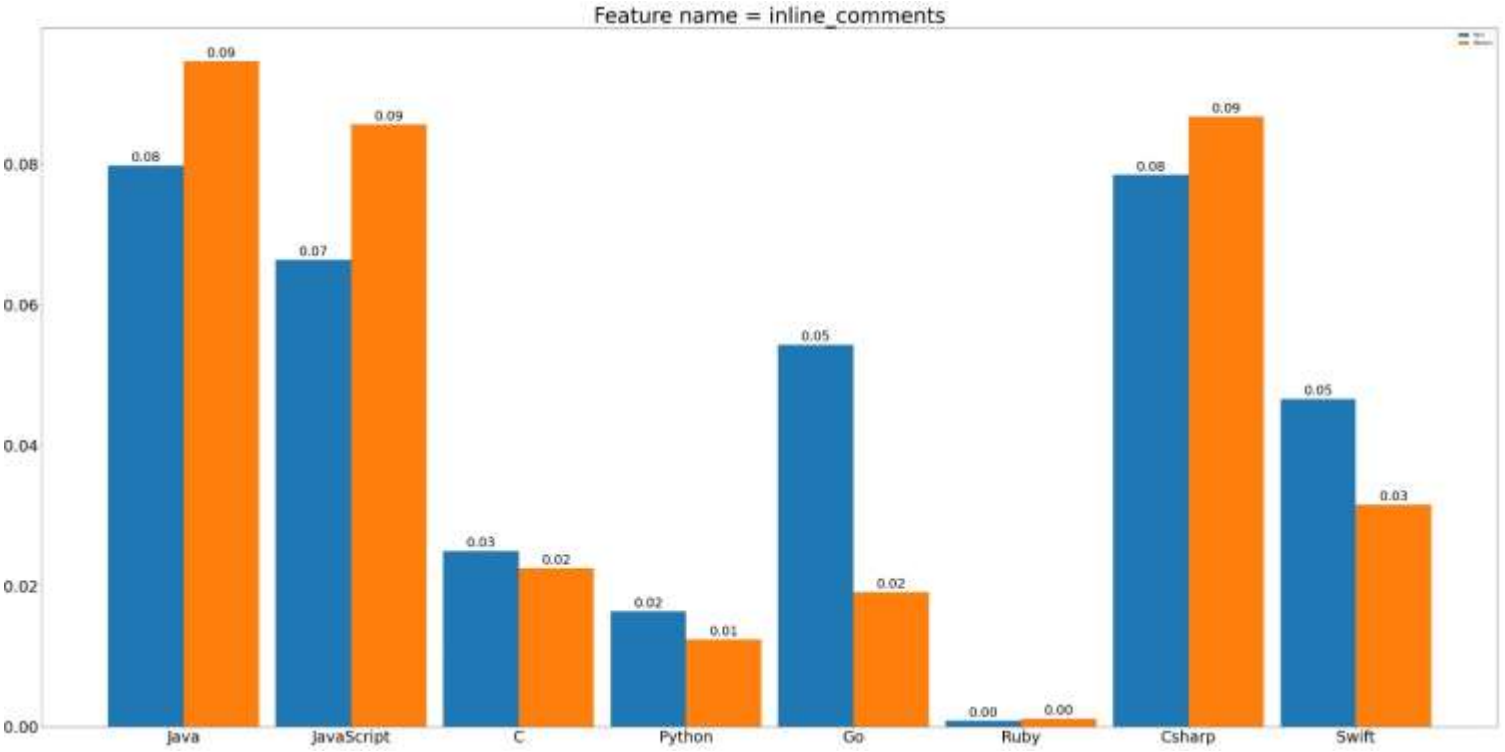


Figure 26. Average number of inline comments by gender

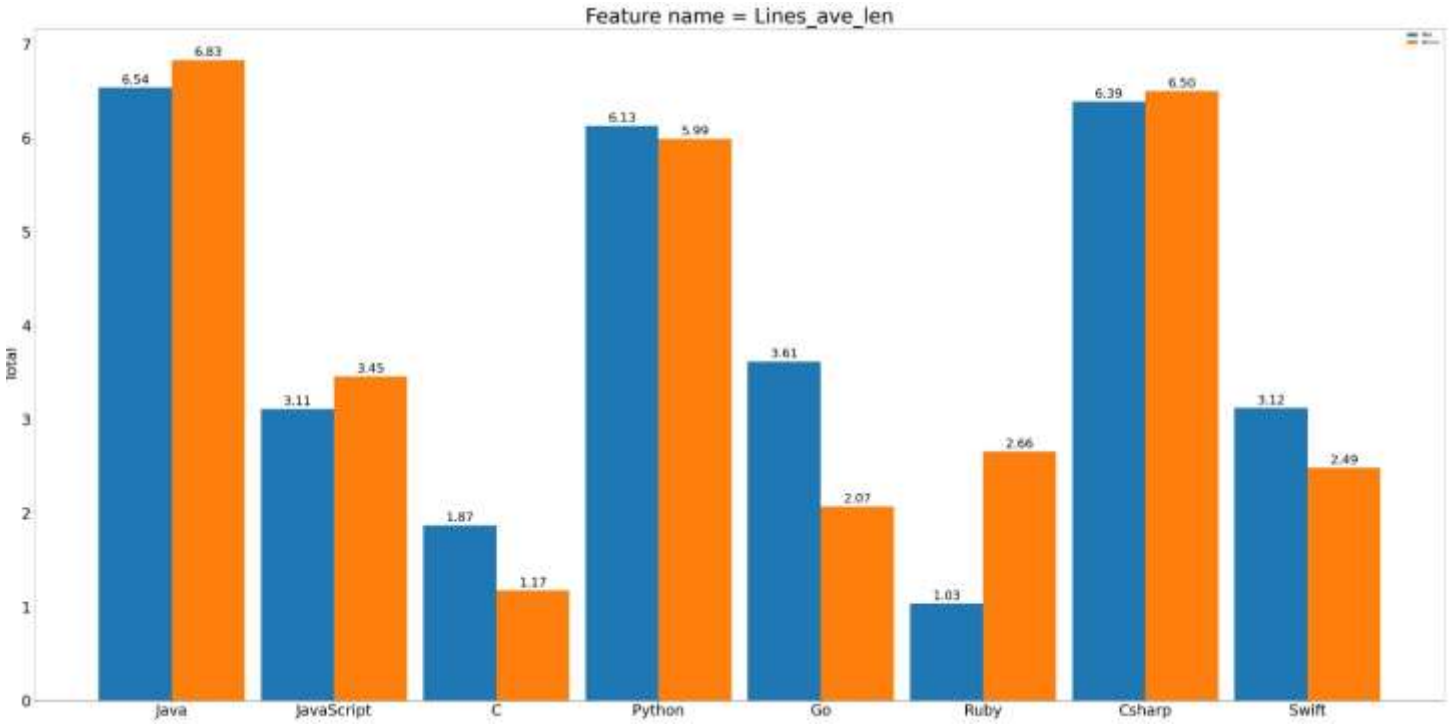


Figure 28. Average lines length by gender

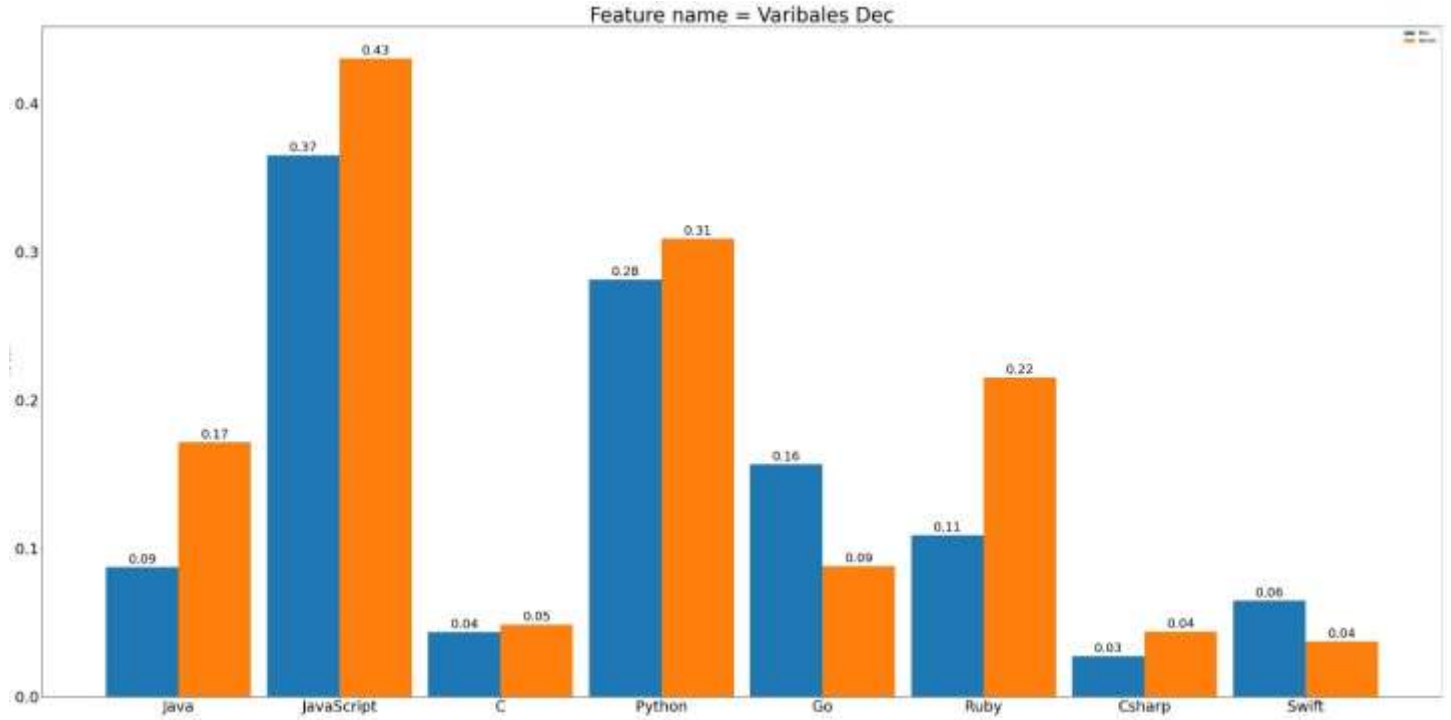


Figure 29. Average Variable's decleration by gender

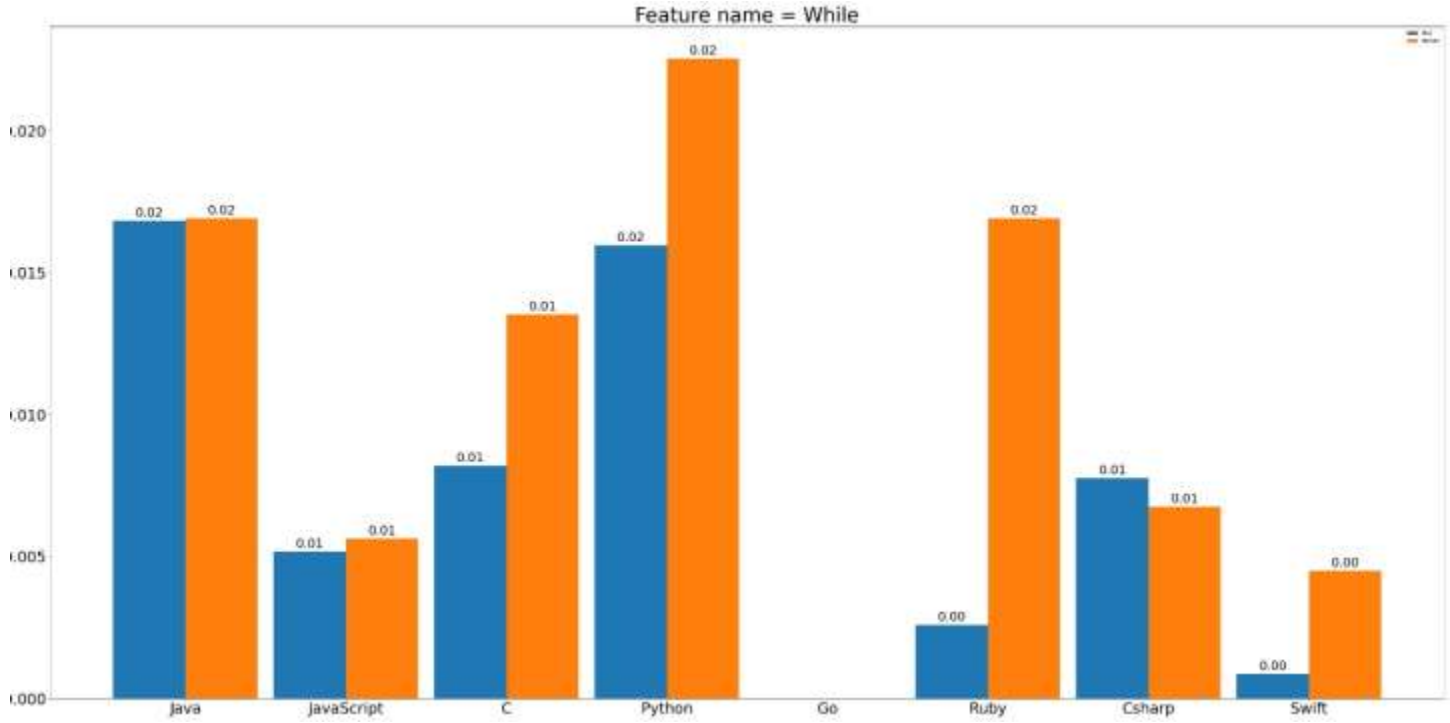


Figure 30. Average number of while loop uses by gender

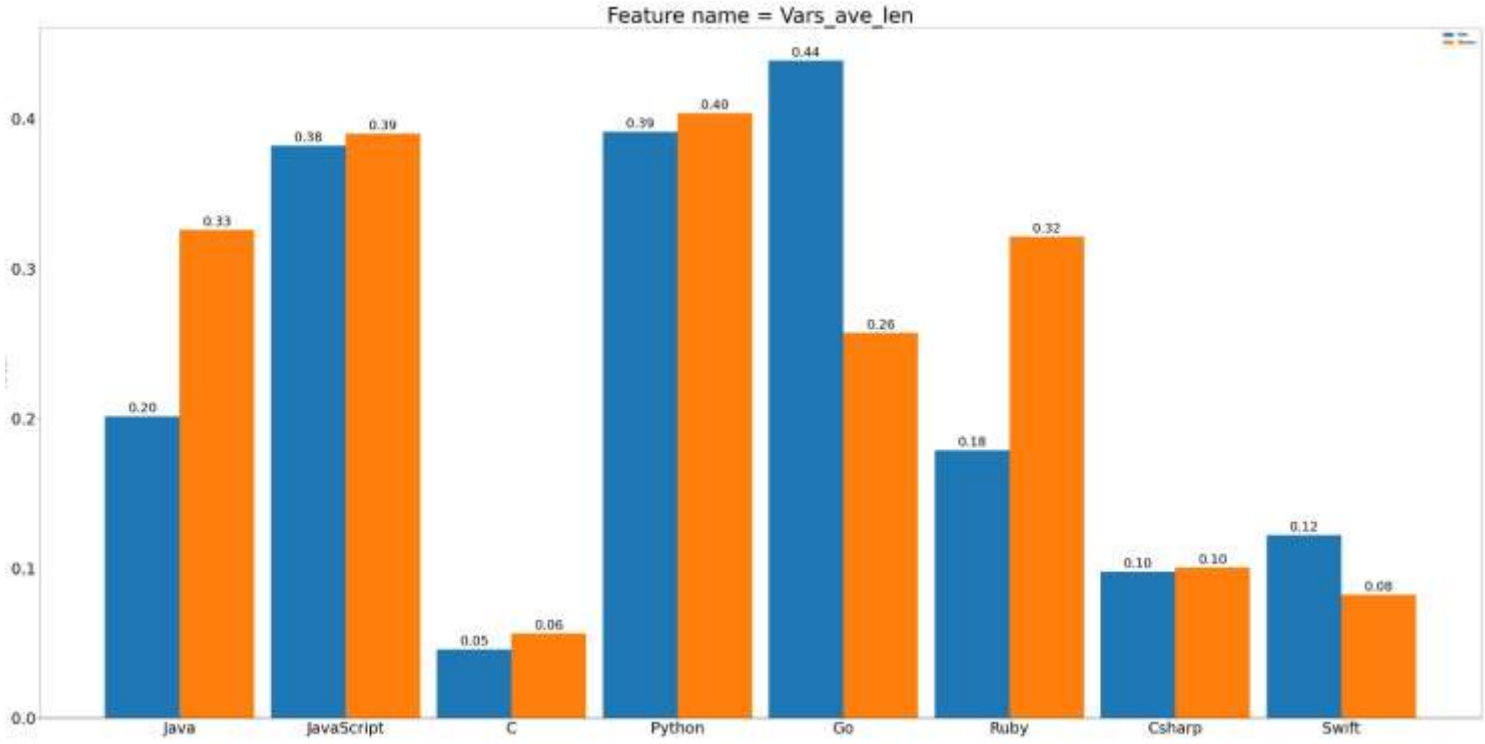


Figure 31. Average variables names length by gender

Modeling Flow:

User_Score	User's_Comments	User_Gender	Name_or_Image	...	value	we	wo	wor	word	word	words
15	0	0	0	...	0	0	0	0	0	0	0
9	0	1	1	...	0	0	0	0	0	0	0
15	0	0	0	...	0	0	0	0	0	0	0
15	0	0	0	...	0	0	0	0	0	0	0
14	0	0	0	...	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...
10	0	0	0	...	0	0	0	0	0	0	0
10	0	0	0	...	0	0	1	1	1	0	1
10	0	0	0	...	0	0	0	0	0	0	0

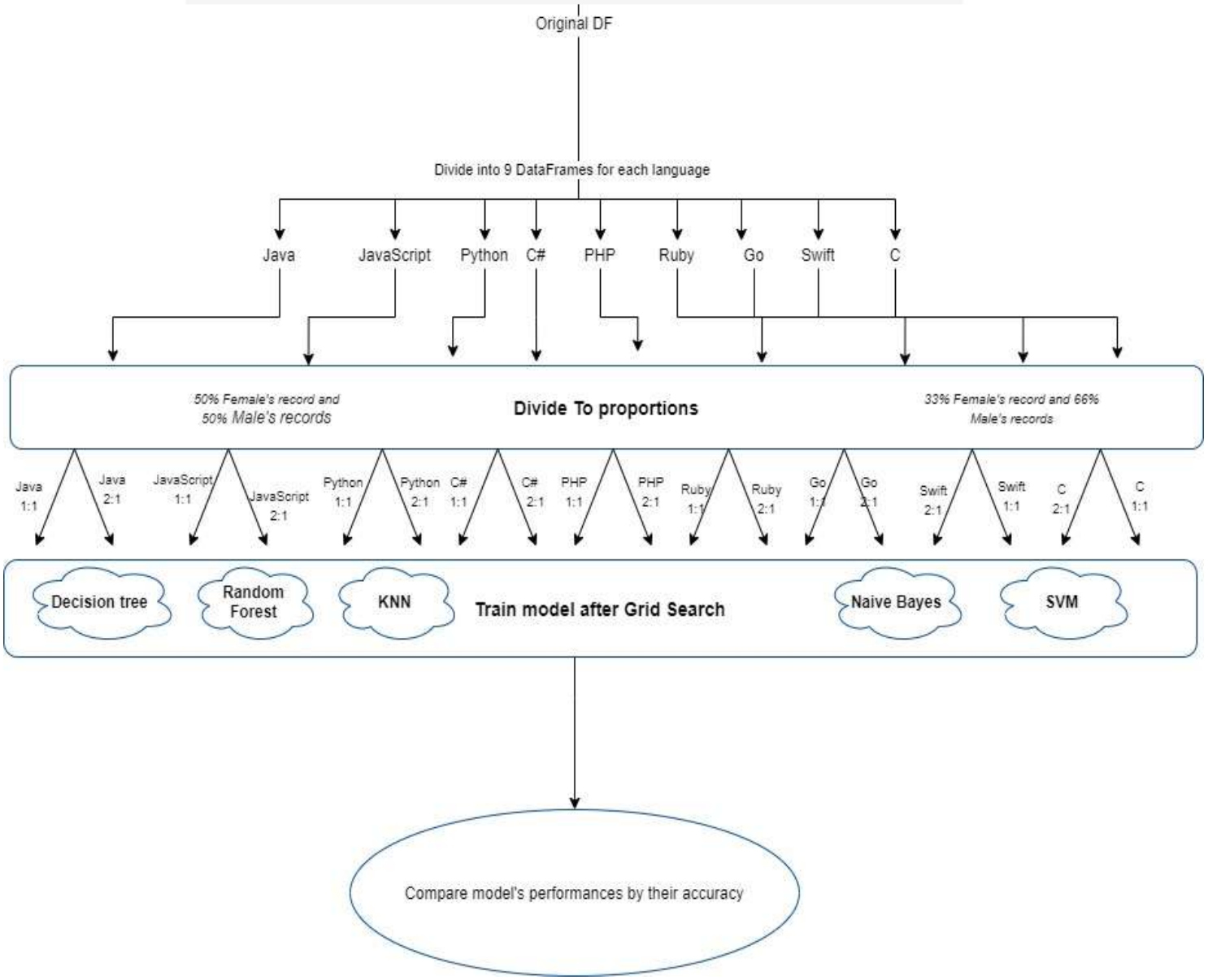


Figure 32. The Modeling Flow

Modeling Flow Description:

The main steps in the modeling stage:

1. Divide the DataFrames by each of the programming languages.

   The total number of codes belonging to men was 2317 while the number of codes written by women was 887, which makes sense for the current programmer population.

2. Because of the unbalanced data – Create two different 'proportion DataFrames' for each of the language dataframe chosen randomly so that for each challenge there will be the same proportion of genders.
   a. The first one is a Dataframe that contains 50% female codes and 50% male codes.
   b. The second contains 33% female codes and 66% male codes.

   Finally, we had to test a total number of 36 DataFrames: Two types of DataFrames that were divided into 9 languages. For each one we tried scaled and non-scaled dataframe.

3. Examine five different machine learning models from the scikit-learn library:
   - Decision Tree
   - Random Forest

- KNN
- Naïve Bayes
- SVM

Test all models above and compare their performances by their accuracy and f1 score rates.

4. In order to find the best-fitted hyper-parameters, use grid search with multiple parameters options and choose the best one.

   "Hyper-parameters are parameters that are not directly learnt within estimators. In scikit-learn they are passed as arguments to the constructor of the estimator classes. Typical examples include C, kernel and gamma for Support Vector Classifier, alpha for Lasso, etc.

   It is possible and recommended to search the hyper-parameter space for the best cross validation score.

   Any parameter provided when constructing an estimator may be optimized in this manner" [16].

```
def find_best_k_for_KNN(X_train, y_train):
    parameters = {'n_neighbors':[10,15,20,23,25]}
    knn = KNeighborsClassifier(n_neighbors=parameters['n_neighbors'])
    clf = GridSearchCV(knn,parameters,scoring=make_scorer(metrics.f1_score, greater_is_better=True))
    clf.fit(X_train, y_train)
    best_K = clf.best_params_['n_neighbors']
    return best_K
```

Figure 33. KNN's Hyper Parameters

```python
def find_best_decision_tree_params(X_train, y_train):

    parameters = {'max_depth':[1,2,3,4,5],"min_samples_split":[5,6,10,12,17,20] }
    dt = tree.DecisionTreeClassifier()

    clf = GridSearchCV(dt, parameters,scoring=make_scorer(metrics.f1_score, greater_is_better=True))
    clf.fit(X_train, y_train)
    best_max_depth = clf.best_params_['max_depth']
    best_min_samples_split = clf.best_params_['min_samples_split']
    return best_max_depth, best_min_samples_split
```

Figure 34. Decision Tree's Hyper Parameters

```python
def find_best_random_forest_num_estimators(X_train, y_train):
    param_grid = {'max_depth':[1,2,3,4,5,6,10],"min_samples_split":[5,10,17,22],'n_estimators':[100, 200, 300]}

    rf = RandomForestClassifier()

    clf = GridSearchCV(rf, param_grid,scoring=make_scorer(metrics.f1_score, greater_is_better=True))
    clf.fit(X_train, y_train)

    best_num_estimators = clf.best_params_

    return best_num_estimators
```

Figure 35. Random Forest's Hyper Parameters

```
def find_best_svm(X_train, y_train):
    param_grid= {'kernel': ('linear', 'rbf'),'C': [0.1, 0.01, 1, 10]}
    sv = svm.SVC()
    clf = GridSearchCV(sv, param_grid,scoring=make_scorer(metrics.f1_score, greater_is_better=True))
    clf.fit(X_train, y_train)

    best_kernel = clf.best_params_['kernel']
    best_C = clf.best_params_['C']

    return best_kernel, best_C
```

Figure 36. SVM's Hyper Parameters

```
def find_best_model(X_train, y_train, max_depth_val, min_samples_split_val,clfs):

    best_recall_val = 0
    best_clf = clf1
    for i,clf in enumerate(clfs):
        print(clf)
        clf.fit(X_train,y_train)
        if(i==2): continue
        scores = cross_val_score(clf, X_train, y_train, cv=10).mean()
        print(scores)
        if(best_recall_val<scores):
            best_clf = clf
            best_recall_val = scores

    return best_clf, best_recall_val
```

Figure 37. Cross validation – find the best scores

After finding the best parameters, we trained and tested the models using two ways of evaluating estimator performance:

1. Cross validation score using cv = 10:

```
scores = cross_val_score(clf, X, y, cv=10)
```

2. Cross validation score using cv = Leave On Out:

```
Loo = cross_val_score(clf, X, y, cv= LeaveOneOut())
```

Results and discussion

After training and testing the models, we summarized the best performance for each language:

Language	Best Model	Best DataSet	Scalling	Best Gender Proportion	Best accuracy
Java	DT LOO	Regular Data	No	**2:1** (66% Male, 33% Female)	81.1%
Csharp	DT, LOO	Normalized	Both	**2:1** (66% Male, 33% Female)	74.8%
JavaScript	DT LOO	Normalized	No	**2:1** (66% Male, 33% Female)	88.6%
Python	SVM LOO	Normalized	Yes	**2:1** (66% Male, 33% Female)	74.8%
Ruby	Both SVM, SVM LOO, and DT LOO	Both regular and norm	Both	**2:1** (66% Male, 33% Female)	89.6%
PHP	NB	Both regular and norm	Both	**2:1** (66% Male, 33% Female)	77.5%
Go	DT LOO	Norm	Yes	**1:1** (50% Male, 50% Female)	78.3%
Swift	DT LOO	Regular	No	**2:1** (66% Male, 33% Female)	77.7%
Average accuracy					**80.3%**

Figure 38. Best Performances table

We noticed that most models rely on the "Image_or_Name" column, (described in page 21) which does not contribute to our research question, but refers to how the data was extracted. Following this, we decided to remove this column and examine the performance of the various models without it:

Language	Best Model	Best DataSet	Scalling	Best Gender Proportion	F1_score_micro	Best accuracy
Java	DT LOO	Regular Data	No	**2:1** (66% Male, 33% Female)	0.625	70%
Csharp	DT LOO	Normalized	No	**2:1** (66% Male, 33% Female)	0.623	74.8%
JavaScript	KNN, KNN LOO	both Normalized and Regular	No	**2:1** (66% Male, 33% Female)	0.651 norm, 0.66 reg	70%
Python	DT LOO	Normalized and regular	both	**2:1** (66% Male, 33% Female)	0.606	71.4 %
Ruby	KNN LOO	Regular	No	**2:1** (66% Male, 33% Female)	0.72	67.7%
PHP	NB, SVM LOO	Norm	Both	**2:1** (66% Male, 33% Female)	0.714 SVM	75.5 %
Go	DT, RF	Norm	both	**2:1** (66% Male, 33% Female)	0.774 RandomForest	76.2%
Swift	DT	Regular	No	**2:1** (66% Male, 33% Female)	0.77	77.8 %
Averages					**0.69**	**73%**

Figure 39. Best performances table without "Image_or_Name" column

As we expected, the performances decreased, but **the models can still predict the user's gender** with an average accuracy of about 73%, and f1_score of about 70%, which is encouraging.

- For all languages, the division by 66% men and 33% women seems to be the most appropriate for predictive purposes.
- For most languages, the decision tree model gained the best performance.
- There seem to be some differences in code writing between male and female, but for each language, the features that influence the differences are not the same.
- In all of the languages beside Java, males used more Capital letters than females did.
- The model performances yielded a good initial result for further research.

Extract Models Coefficients by language:

Java

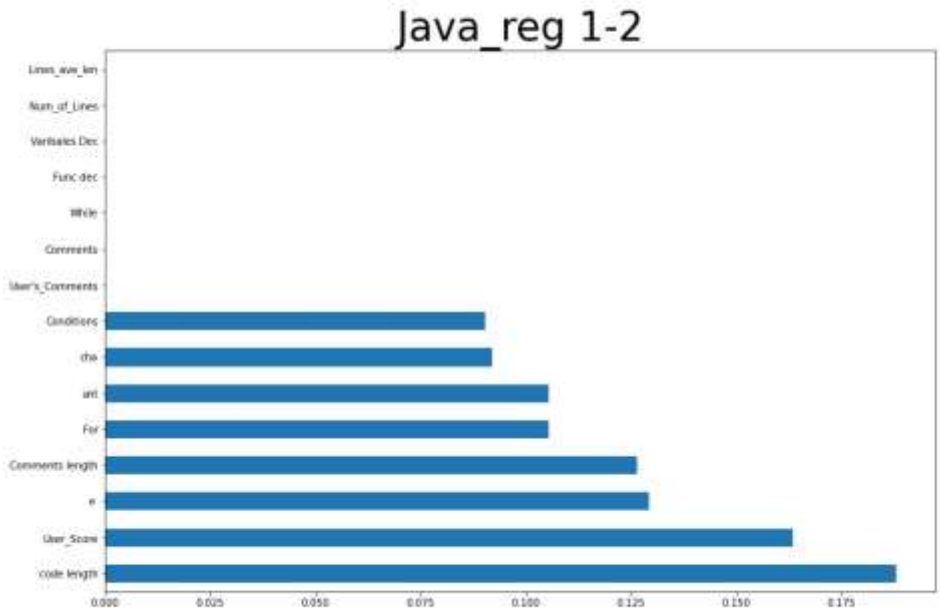


Figure 40 Java coefs

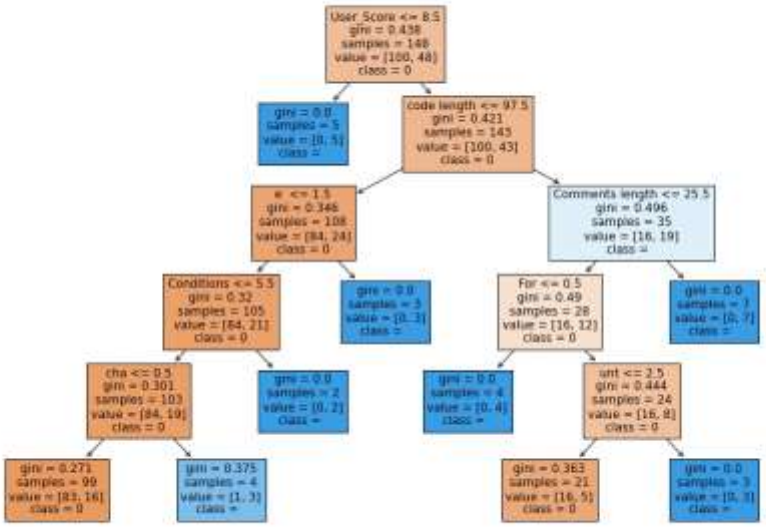


Figure 41 Java decision tree nodes

Csharp

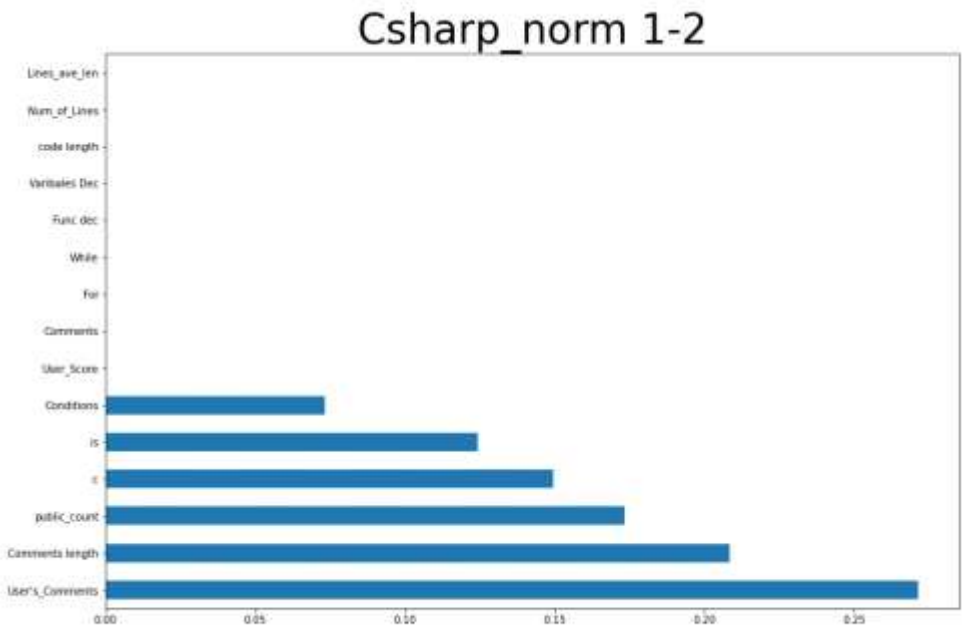


Figure 42 Csharp coefs

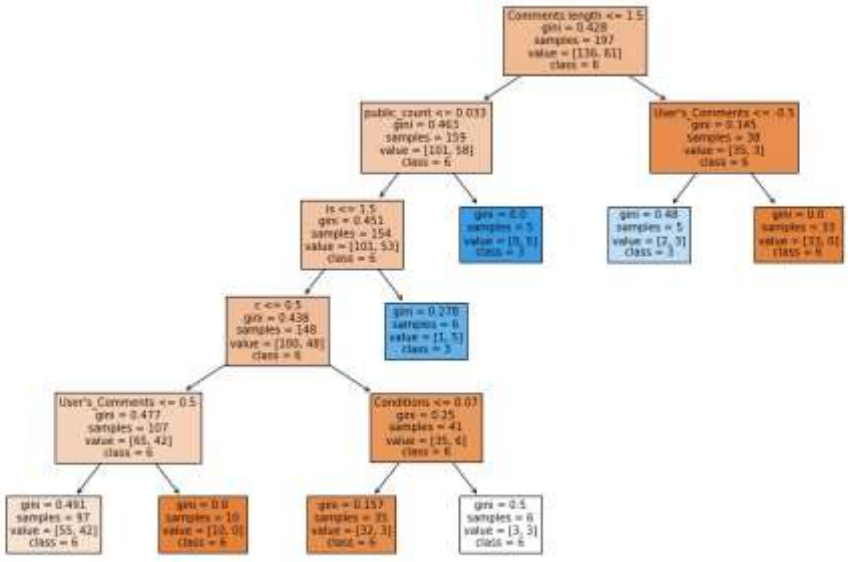


Figure 43 Csharp decision tree nodes

JavaScript

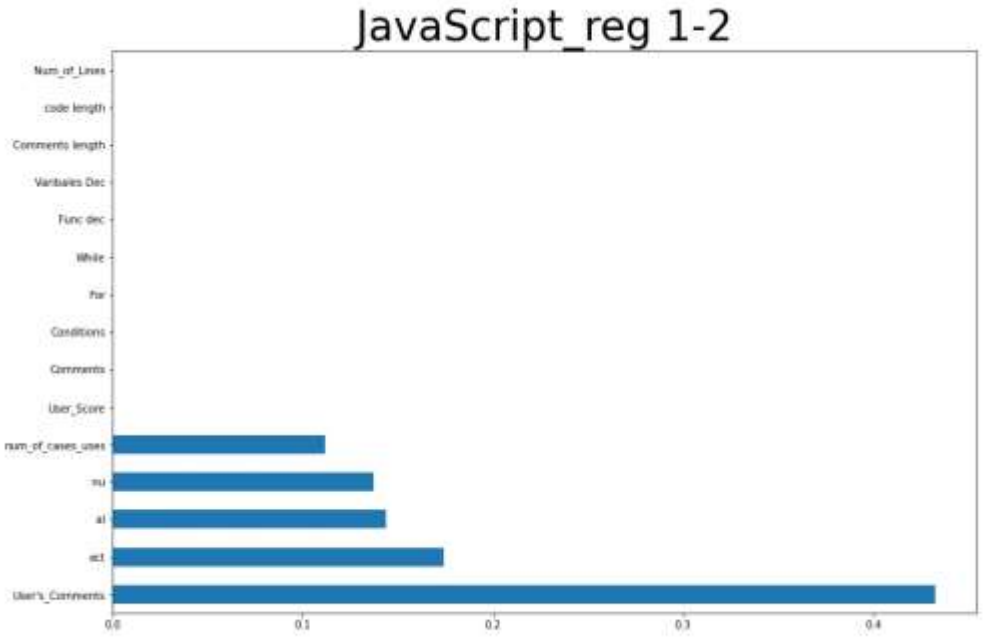


Figure 44 Java Script coefs

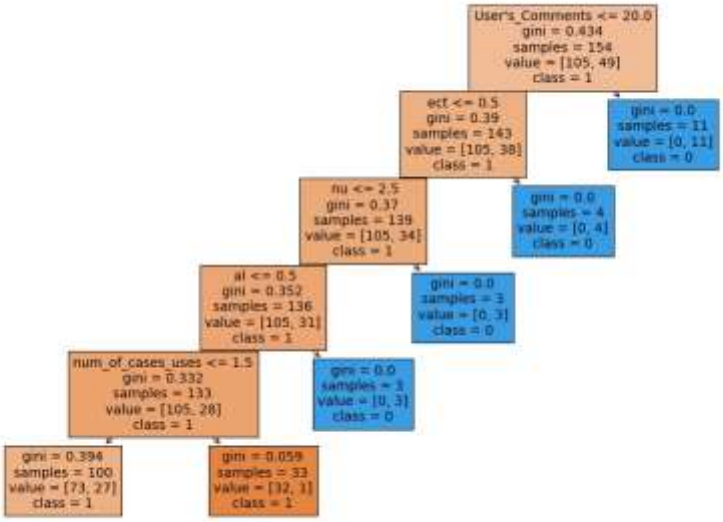


Figure 45. JavaScript decision tree nodes

Python

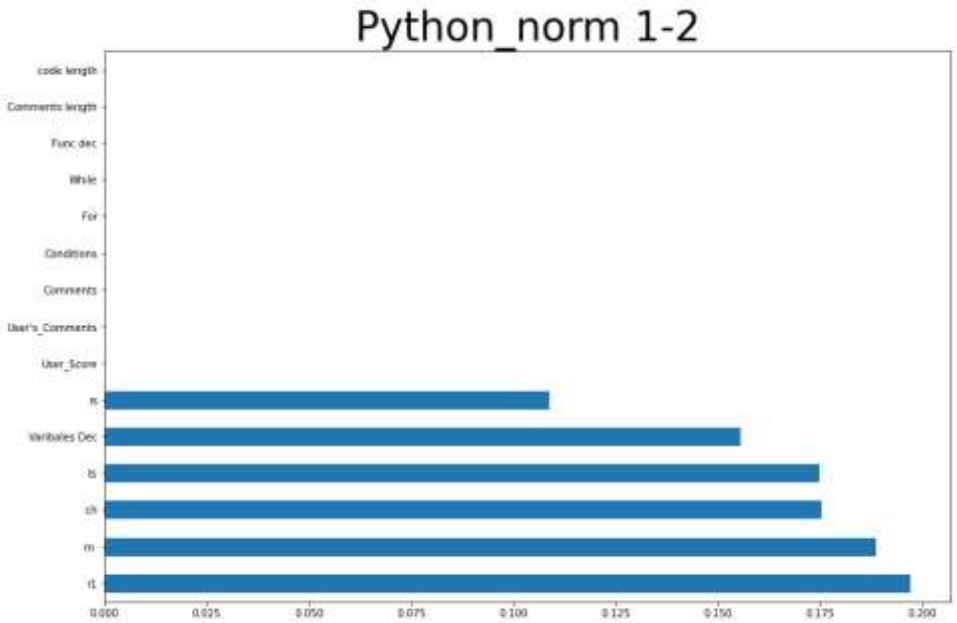


Figure 46. Python coefs

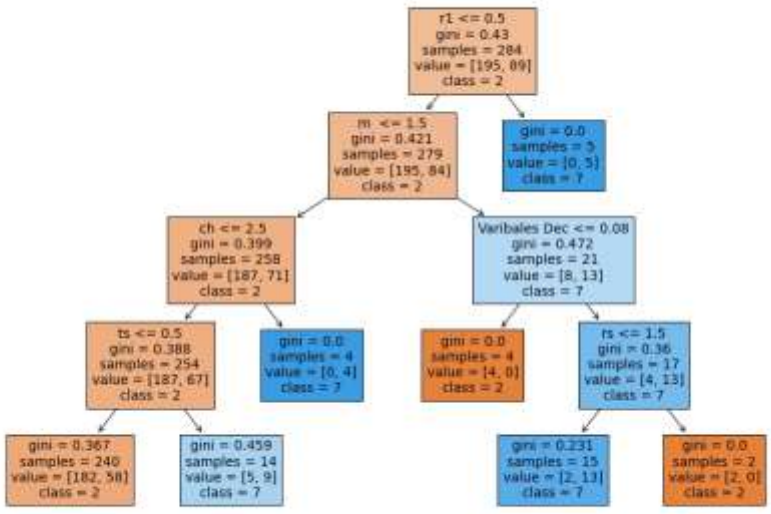


Figure 47. Python decision tree nodes

Ruby

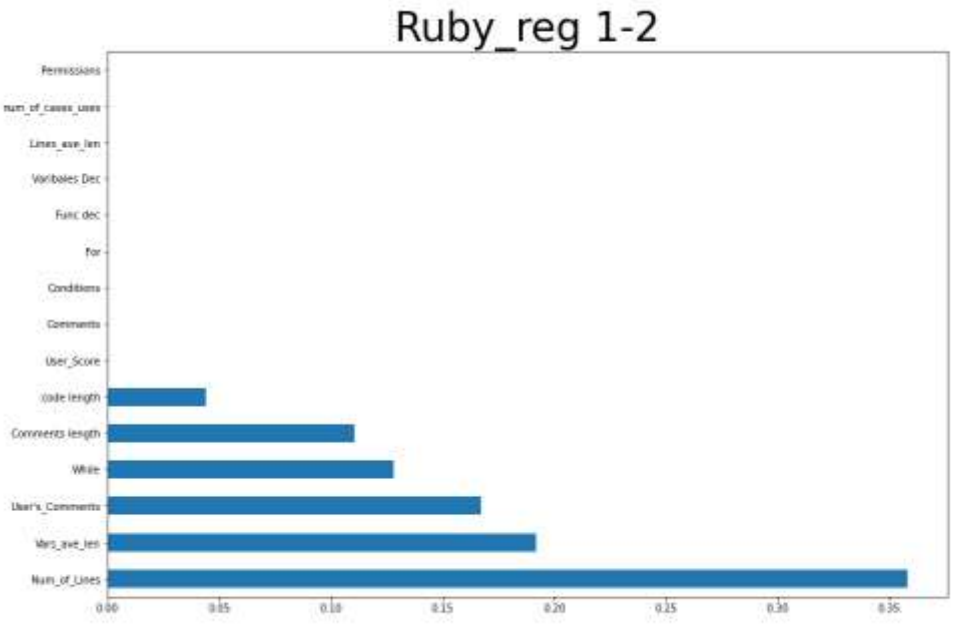


Figure 48. Ruby coefs

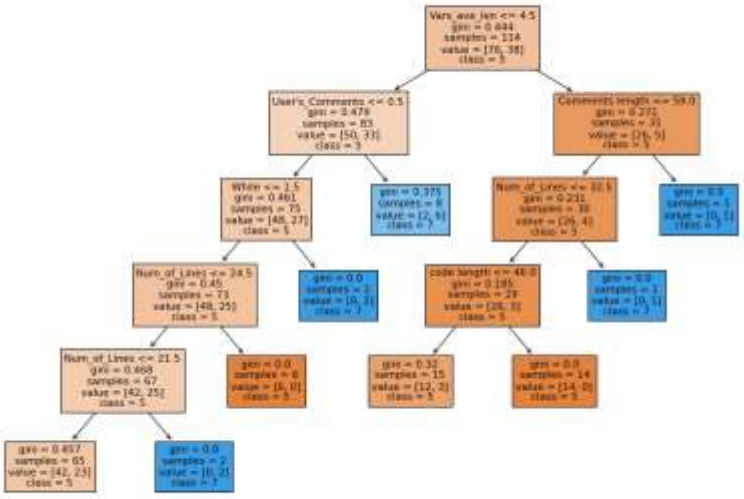


Figure 49. Ruby decision tree nodes

PHP

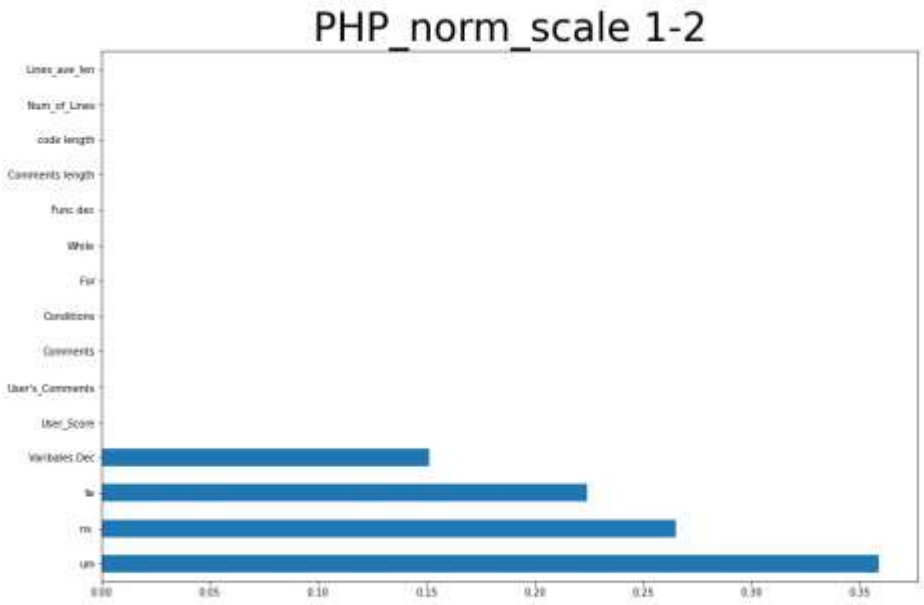


Figure 50. PHP coefs

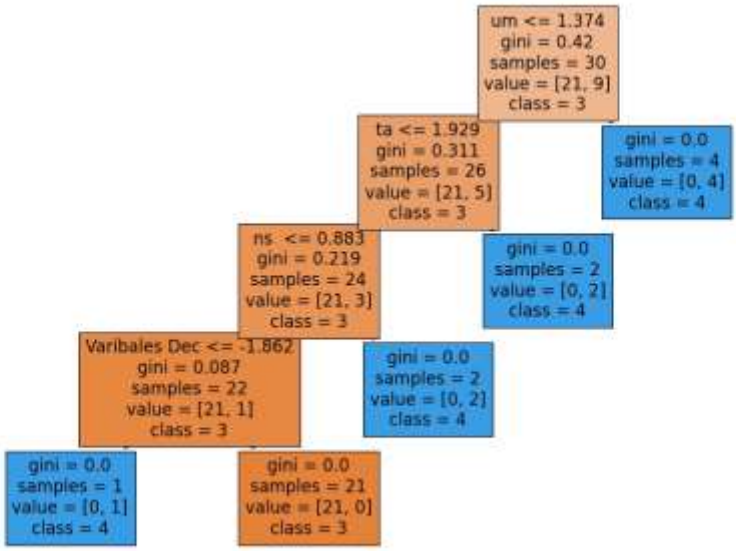


Figure 51. PHP decision tree nodes

Go

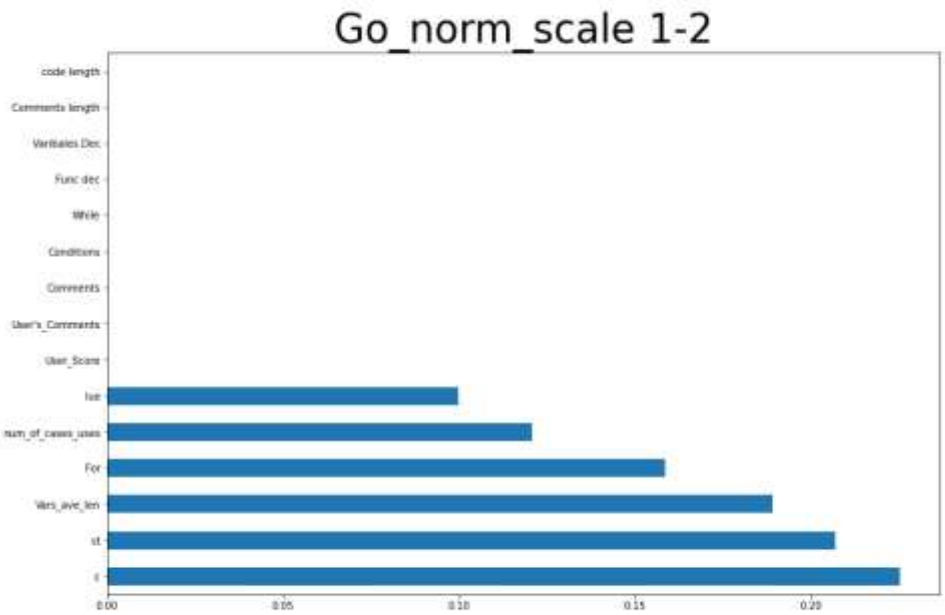


Figure 52. Go Coefs

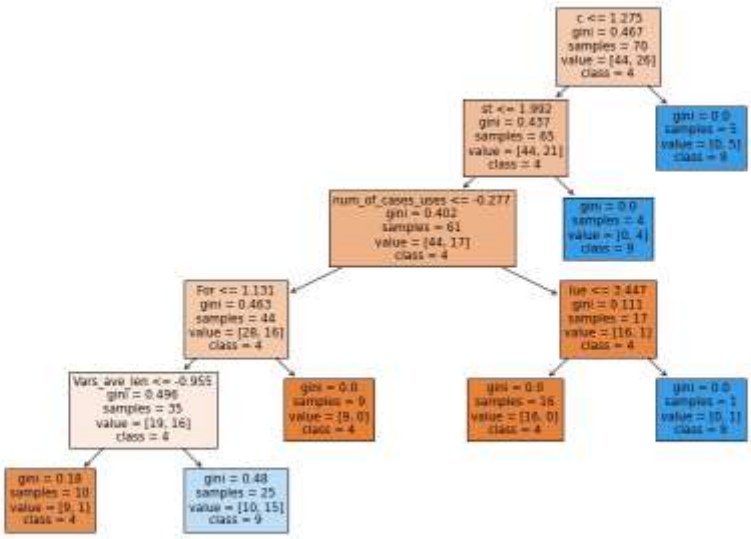


Figure 53. Go decision tree nodes

Swift

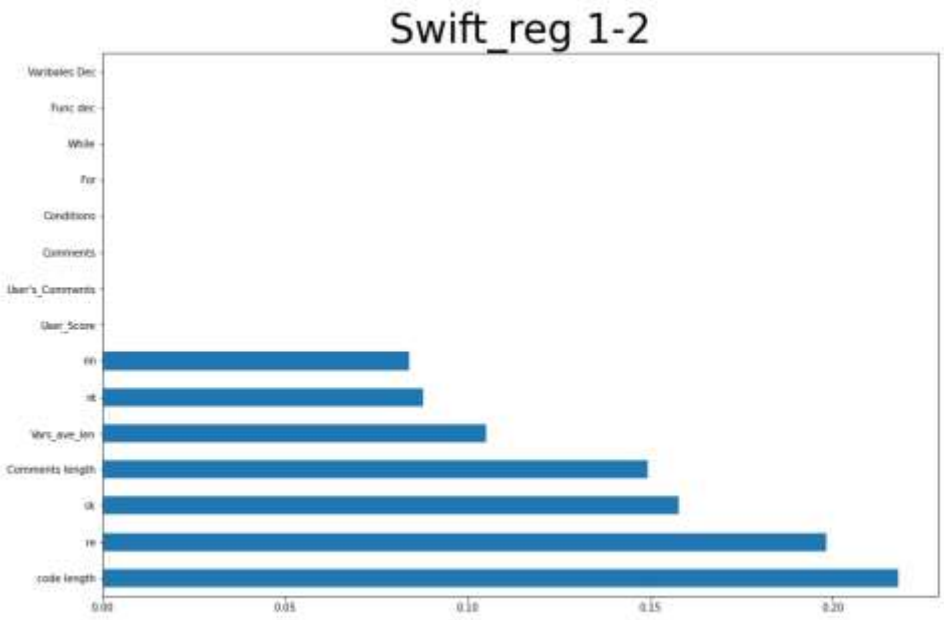


Figure 54. Swift coefs

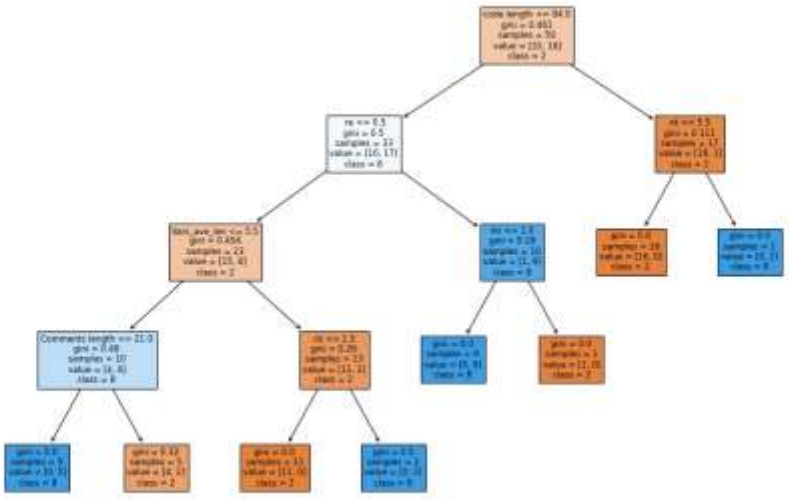


Figure 55. Swift decision tree

Conclusions and future research recommendations:

Our Research question was - "Can we predict the programmer's gender from a given code?"

We think that our project shows a promising start in the direction of detecting the gender from a given programming code. A strong proof for this assumption can be seen through the rating of the models - which means they actually learned and did not guess. This shows that we succeeded in our major challenge, which was - translating the programming code into countable and meaningful features, so that we can distinct between females and males.

The fact that we initially selected unlabeled data, which means we did not know the genders of the writers for sure, may be one of the problematic factors for the results. This fact also forced us to use less data records than we could have used, because most of the users did not give their real name, or the detector could not detect their gender.

In order to answer the important and more detailed question – 'how well can a machine detect the gender of the writer from a given code?'- further work should include more research about each

language in order to find more features along with using gender labeled or supervised data.

Figures

Bibliography

[1] https://docs.anaconda.com/ae-notebooks/user-guide/basic-tasks/apps/jupyter/

[2] https://en.wikipedia.org/wiki/Python_(programming_language)

[3] https://www.pathrise.com/guides/a-review-of-coderbyte-as-a-software-engineer-interview-prep-tool/

[4] https://en.wikipedia.org/wiki/Pandas_(software)

[5] https://numpy.org/doc/stable/user/whatisnumpy.html

[6] https://seaborn.pydata.org/introduction.html

[7] https://opencv.org/about/

[8] https://www.activestate.com/resources/quick-reads/what-is-matplotlib-in-python-how-to-use-it-for-plotting/

[9] https://www.guru99.com/introduction-to-selenium.html

[10] https://programminghistorian.org/en/lessons/intro-to-beautiful-soup#what-is-beautiful-soup

[11] https://autohotkey.com/board/topic/20260-gender-verification-by-forename-cmd-line-tool-db/

[12] https://en.wikipedia.org/wiki/Scikit-learn

[13] https://www.tutorialspoint.com/python/python_reg_expressions.htm

[14] https://www.guru99.com/scipy-tutorial.html

[15] https://www.geeksforgeeks.org/using-countvectorizer-to-extracting-features-from-text/

[16] https://scikit-learn.org/stable/modules/grid_search.html