Code Smell Detection

In this Notebook we're going to use https://zenodo.org/record/3590102#.YvEOaHZByUk datasets to train a ML algorithm for smell prediction.

In the first steps we import our Pandas module to handle the datasets and its modification.

In [40]:

```
import pandas as pd

# Read the csv file
df = pd.read_csv('D:\\Articles\\Code Smells\\CodeSmells\\Smell Project\\Datas

# Print it out if you want
print(df.shape, "\n")
print("This dataset consists of following features:\n", list(df.columns), sep
df
```

```
(14853, 15)

This dataset consists of following features:
['id', 'reviewer_id', 'sample_id', 'smell', 'severity', 'review_timestamp',
'type', 'code_name', 'repository', 'commit_hash', 'path', 'start_line', 'en
d_line', 'link', 'is_from_industry_relevant_project']
```

Out[40]:

	id	reviewer_id	sample_id	smell	severity	review_timestamp	type	
0	526	6	5771277	feature envy	none	34:53.0	function	org.apache
1	527	6	5771277	long method	none	34:53.0	function	org.apache
2	528	6	5786929	blob	critical	37:38.1	class	org.apa
3	529	6	5786929	data class	critical	37:38.1	class	org.apa
4	530	6	5788107	feature envy	none	37:49.6	function	org.apacl
...	...	...	...	...	...	...	...	
14848	15348	20	8968865	blob	none	24:46.5	class	
14849	15349	20	9474095	data class	none	24:54.3	class	org.apac
14850	15350	20	9474095	blob	none	24:54.3	class	org.apac
14851	15351	20	6293245	data class	none	25:13.0	class	org.e
14852	15352	20	6293245	blob	minor	25:13.0	class	org.e

14853 rows × 15 columns

In [41]:

```
df["severity"].unique()
```

Out[41]:

```
array(['none', 'critical', 'minor', 'major'], dtype=object)
```

```
There are too many none value for severity feature. According to the art
icle value 'none' for severity means that none of the code reviewers kno
w whether the code has smell or not so in the next cell we're going to d
elete the rows containing none value for severity as they are useless in
```

```
our computations. However we can use these records as a new data that we
want to predict So in the future they maybe usefull for us.
```

In [42]:

```
dfWithoutNone = df[df['severity'] != "none"]
dfWithoutNone.to_csv("D:\\Articles\\Code Smells\\CodeSmells\\Smell Project\\D
dfWithoutNone.shape
```

Out[42]:

```
(3301, 15)
```

In [43]:

```
dfWithoutNone
```

Out[43]:

	id	reviewer_id	sample_id	smell	severity	review_timestamp	type	
2	528	6	5786929	blob	critical	37:38.1	class	org.apach
3	529	6	5786929	data class	critical	37:38.1	class	org.apach
25	551	6	5822090	blob	minor	42:50.3	class	(
28	554	6	5828468	blob	major	44:23.9	class	org.apache
29	555	6	5828468	data class	minor	44:23.9	class	org.apache
...	...	...	...	...	...	...	...	
14835	15335	20	7916535	long method	major	22:27.6	function	org.apacl
14838	15338	20	8169113	data class	major	22:46.3	class	org.a
14843	15343	20	7638207	blob	minor	24:31.2	class	org.sprin(
14847	15347	20	8968865	data class	major	24:46.5	class	
14852	15352	20	6293245	blob	minor	25:13.0	class	org.eclip

3301 rows × 15 columns

```
Let's seprate and categorize the type of smells existing in this datase
t. With the following code we can get the varies of the smell in this da
taset and it is 4 different smell as we expected.
```

In [44]:

```
dfWithoutNone["smell"].unique()
```

Out[44]:

```
array(['blob', 'data class', 'long method', 'feature envy'], dtype=object)
```

```
In the following we are going to seprate these smells and data and addin
```

```
g them to their specified datasets.
```

In [45]:

```
dfFeatureEnvy = dfWithoutNone[dfWithoutNone["smell"] == 'feature envy']
dfFeatureEnvy.to_csv("D:\\Articles\\Code Smells\\CodeSmells\\Smell Project\\D
dfFeatureEnvy
```

Out[45]:

	id	reviewer_id	sample_id	smell	severity	review_timestamp	type	
44	570	7	5839980	feature envy	minor	46:31.2	function	org.apa
47	573	6	5855589	feature envy	minor	47:09.4	function	org.apacl
85	613	6	5922177	feature envy	minor	51:45.2	function	org.apach
94	622	6	5939954	feature envy	major	55:26.3	function	org.apache.
103	631	6	6133230	feature envy	minor	56:52.6	function	org
...	...	...	...	...	...	...	...	
14770	15270	21	6863919	feature envy	minor	29:24.1	function	com.gc
14782	15282	21	8861277	feature envy	minor	35:28.3	function	org.ap
14811	15311	21	4081252	feature envy	critical	40:45.1	function	org.apa
14818	15318	21	8389037	feature envy	minor	54:22.8	function	org.apacl
14820	15320	21	3942097	feature envy	major	54:32.0	function	org.apa

454 rows × 15 columns

In [46]:

```
dfBlob = dfWithoutNone[dfWithoutNone["smell"] == 'blob']
dfBlob.to_csv("D:\\Articles\\Code Smells\\CodeSmells\\Smell Project\\Datasets
dfBlob
```

Out[46]:

	id	reviewer_id	sample_id	smell	severity	review_timestamp	type	
2	528	6	5786929	blob	critical	37:38.1	class	org.apache
25	551	6	5822090	blob	minor	42:50.3	class	oı
28	554	6	5828468	blob	major	44:23.9	class	org.apache.ι
64	592	6	5884892	blob	major	50:05.9	class	org.apache.ι
86	614	6	5938889	blob	minor	53:01.7	class	org.apache.z
...	...	...	...	...	...	...	...	
14827	15327	20	5247450	blob	minor	17:04.7	class	org.apa
14829	15329	20	6454251	blob	minor	17:22.0	class	oı
14834	15334	20	8528735	blob	major	22:18.9	class	com.cloud.netw
14843	15343	20	7638207	blob	minor	24:31.2	class	org.springf
14852	15352	20	6293245	blob	minor	25:13.0	class	org.eclips

984 rows × 15 columns

```
In [47]:  dfDataClass = dfWithoutNone[dfWithoutNone["smell"] == 'data class']
          dfDataClass.to_csv("D:\\Articles\\Code Smells\\CodeSmells\\Smell Project\\Dat
          dfDataClass
```

Out[47]:

	id	reviewer_id	sample_id	smell	severity	review_timestamp	type	
3	529	6	5786929	data class	critical	37:38.1	class	org.apache.te
29	555	6	5828468	data class	minor	44:23.9	class	org.apache.un
34	560	7	5827650	data class	minor	44:58.5	class	org.apache.uima
91	619	7	5935402	data class	major	53:35.8	class	org.a
99	627	6	6002683	data class	major	55:55.7	class	org
...	...	...	...	...	...	...	...	
14768	15268	21	6293245	data class	major	56:47.9	class	org.eclipse.
14784	15284	21	6483071	data class	major	35:43.9	class	
14800	15300	21	4503985	data class	minor	38:39.1	class	org.apache.flu
14838	15338	20	8169113	data class	major	22:46.3	class	org.apac
14847	15347	20	8968865	data class	major	24:46.5	class	

1057 rows × 15 columns

In [48]:

```
dfLongMethod = dfWithoutNone[dfWithoutNone["smell"] == 'long method']
dfLongMethod.to_csv("D:\\Articles\\Code Smells\\CodeSmells\\Smell Project\\Da
dfLongMethod
```

Out[48]:

	id	reviewer_id	sample_id	smell	severity	review_timestamp	type	
39	565	6	5840527	long method	minor	45:41.3	function	org.apach
84	612	6	5922177	long method	minor	51:45.2	function	org.apach
95	623	6	5939954	long method	critical	55:26.3	function	org.apache
152	682	7	6182357	long method	minor	08:50.9	function	org
180	710	7	6236496	long method	major	17:30.9	function	org.ec
...	...	...	...	...	...	...	...	
14769	15269	21	6863919	long method	minor	29:24.1	function	com.go
14817	15317	21	8389037	long method	minor	54:22.8	function	org.apac
14819	15319	21	3942097	long method	minor	54:32.0	function	org.ap
14826	15326	20	8861277	long method	minor	16:50.9	function	org.ap
14835	15335	20	7916535	long method	major	22:27.6	function	org.ap

806 rows × 15 columns

In the next step we are going to download each of these datasets(smells) refrence code. Refrence code has been determined in the mentioned articles dataset in the link feature. In the following we have written a function that takes a url as input and finds the code and saves it into a file.

In [49]:

```
import requests
import lxml
from bs4 import BeautifulSoup

def getCodes(url, dframe, pathToSave, counter):
    global errorLinks
    headers = {
      'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (
    }
    req = requests.get(url, headers = headers)
    soup = BeautifulSoup(req.content,'lxml')
    code_id = list(dframe['id'])[counter]

    startLine = list(dframe["start_line"])[counter]
    EndLine = list(dframe["end_line"])[counter] # determined in the dataset

    file = open(f"{pathToSave}\\row-{counter}-id{code_id}.java", "w")
    t = True
    for index in range(startLine, EndLine+1):
        if t:
            t = False
            pass
        else:
            file.write("\n")
        try:
            line = soup.find('td', {'id':f'LC{index}'}).get_text()
            file.write(line)
        except AttributeError:
            errorLinks.append(url)
            file.close()
            return
        except:
            errorLinks.append([url, "Other Errors"])
            file.close()
            return
    file.close()
```

In [50]:

```
# Defines with CSV
errorLinks = list()

pathToSaveCodeBlob = "D:\\Articles\\Code Smells\\CodeSmells\\Smell Project\\D
pathToSaveCodeDataClass = "D:\\Articles\\Code Smells\\CodeSmells\\Smell Proje
pathToSaveCodeLongMethod = "D:\\Articles\\Code Smells\\CodeSmells\\Smell Proj
pathToSaveCodeFeatureEnvy = "D:\\Articles\\Code Smells\\CodeSmells\\Smell Pro

# Blob Smell # # # # # # # # # # # DONE # # # # # # # # # # # #
# Crawling Blob's Code from the mentioned link in the dataset
# cnt = 629
# for linkindex in range(629, len(dfBlob['link'])):
#     getCodes(list(dfBlob['link'])[linkindex], dfBlob, pathToSaveCodeBlob, c
#     cnt += 1
#     if cnt == 53:
#         break

    # To add
    ## We must delete the codes that aren't available and we couldn't find th
# Blob Smell # # # # # # # # # DONE # # # # # # # # # # # # # #


# DataClass Smell # # # # # # # # # # # DONE # # # # # # # # # # # #
# Crawling DataClass's Code from the mentioned link in the dataset
# cnt = 0
# for linkindex in range(0, len(dfDataClass['link'])):
#     getCodes(list(dfDataClass['link'])[linkindex], dfDataClass, pathToSaveC
#     cnt += 1
#     if cnt == 53:
#         break

    # To add
    ## We must delete the codes that aren't available and we couldn't find th
# DataClass Smell # # # # # # # # # DONE # # # # # # # # # # # # # #


# DataClass Smell # # # # # # # # # # # DONE # # # # # # # # # # # #
# Crawling LongMethod's Code from the mentioned link in the dataset
# cnt = 0
# for linkindex in range(0, len(dfLongMethod['link'])):
#     getCodes(list(dfLongMethod['link'])[linkindex], dfLongMethod, pathToSav
#     cnt += 1
#     if cnt == 53:
#         break

    # To add
    ## We must delete the codes that aren't available and we couldn't find th
# LongMethod Smell # # # # # # # # # DONE # # # # # # # # # # # # # #


# FeatureEnvy Smell # # # # # # # # # # # DONE # # # # # # # # # # # #
# Crawling FeatureEnvy's Code from the mentioned link in the dataset
# cnt = 0
# for linkindex in range(0, len(dfFeatureEnvy['link'])):
#     getCodes(list(dfFeatureEnvy['link'])[linkindex], dfFeatureEnvy, pathToS
#     cnt += 1
#     if cnt == 53:
#         break

    # To add
    ## We must delete the codes that aren't available and we couldn't find th
# FeatureEnvy Smell # # # # # # # # # # DONE # # # # # # # # # # # # # #
```

Modeling

Two approach:

1. Use each of the smell to train a model which can predict us the severity(The measure of the smell) of the smell
2. Use smell to train an algorithm which can detect the smell in a code whether it's a blob or feature envy or etc smell.

Approach one - How much is the smell ?

Now we're going to use these codes as our training set records and create a model to learn this smell. First we're working on blob smell.

Our severity generally can have three values ['critical', 'minor', 'major'] so each of the code can be in only one of these three categories. Don't forget that we're working on the blob smell.

In [51]:

```
## Defining Train and Test sets
## ??
```

In [56]:

```
# from sklearn.svm import svc
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBRegressor



## Use traing set to train your model
xgb = XGBRegressor(colsample_bytree = 1, learning_rate = 0.1, max_depth = 4,
estimators = {
    'RndmForestClassifier': RandomForestClassifier(),
    'DcsionTreeClassifier': DecisionTreeClassifier(),
    'KNeighborsClassifier': knn,
    'LinearRegression': LinearRegression(),
    'GaussianNB': GaussianNB(),
    'XGBRegressor': xgb
}


for estimator_name, estimator_object in estimators.items():
    kfold = KFold(n_splits=10, random_state=11, shuffle=True)

    scores = cross_val_score(estimator=estimator_object, X=digits.data, y=dig

    print(f"{estimator_name:>20}: " +
         f"mean accuracy = {scores.mean():.2%} " +
         f"standard deviation = {scores.std():.2%}", flush=True)
```

In [ ]:

```
```

Approach Two - Which smell ?

First we need to extract feature. Read below link https://towardsdatascience.com/feature-extraction-techniques-d619b56e31be#:~:text=Feature%20Extraction%20aims%20to%20reduce,the%20original%20set%20of%20features.

In [ ]:

```
## Define Train and Test sets


## Use traing set to train your model

    ## First we need to extract our features

    ## We use these feature and their value and say these are the feature and


## Use Test set To check the accuracy
```