Machine Learning Coding & Classification
ECOICOP classification
Code and data

Krystyna Piątkowska
Marta Kruczek-Szepel, Claude Julien (UNECE)
Agenda

1. Learning and training – statswiki UNECE:
   1.1. ECOICOP dataset
   1.2. Machine Learning tutorial

2. Studies and Code – statswiki UNECE:
   Github files
   2.1. Hyperparameter tuning „for” loop – input/output
   2.2. Hyperparameter tuning GridsearchCV - input/output
   2.3. Best parameters & results

3. User's experiences with the ML code and data shared
1. Learning and training – statswiki UNECE

1.1. ECOICOP dataset
1.2. ML tutorial

Link statswiki:
https://statswiki.unece.org/display/ML/Learning+and+training
2. Studies and Code – statswiki UNECE:

Statswiki link:

https://statswiki.unece.org/pages/viewpage.action?spaceKey=ML&title=Studies+and+Codes

<table>
<thead>
<tr>
<th>6</th>
<th>Coding &amp; Classification</th>
<th>Production description to ECO-COP</th>
<th>Poland</th>
<th>Web scraping data</th>
<th>Naive bayes, Logistic regression, Random forest, Support vector machine, Neural network</th>
<th>Yes (Click Github link)</th>
<th>Python</th>
</tr>
</thead>
</table>

Github source code:

https://github.com/statisticspoland/ecoicop_classification
2.1. Github Files
Hyperparameter tuning „for” loop – input/output

- Linear_SVC
- Logistic_Regression
- Naive_Bayes
- Random_Forest

**logistic_regression.py**

**random_forest.py**

Split to train, validation, test dataset

```python
vectorizer = CountVectorizer()
vectorizer.fit() vectorizer.transform()
```

**Statistical Office in Poznan**
2.2. Github Files

Hyperparameter tuning GridSearchCV – input/output

- Linear_SVC
- Logistic_Regression
- Naive_Bayes
- Random_Forest

naive_bayes_params_tunn.py
linear_SVC_params_tunn.py

Split to train, validation, test dataset

```python
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import LinearSVC
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score

linSVC_pipeline = Pipeline([(('vect', TfidfVectorizer(max_df=0.1, ngram_range=(1, 2), stop_words=stop_words_list, sublinear_tf=True, token_pattern='w+|\[1-9]\.[1-9]|\[1-9]\,\[1-9]|\[1-9]\%|\[1-9]\,\[1-9]|\[1-9]\%'), 'tfidf', TfidfTransformer(norm='l2', smooth_idf=True, sublinear_tf=False, use_idf=True)), ('clfLin', svm.LinearSVC(dual=False, max_iter=1200))])
linSVC_pipeline.fit(X, y)

gs = GridSearchCV(linSVC_pipeline, grid_params, cv=5, n_jobs=-1, verbose=1, error_score=0, scoring='accuracy')
gs = gs.fit(X, y)
```

grid_params = {
    'clfLin__penalty': ('l1', 'l2'),
    'clfLin__multi_class': ('ovr', 'crammer_singer'),
    'clfLin__C': (0.01, 0.1, 1, 10, 100, 1000),
    'clfLin__loss': ('hinge', 'squared_hinge'),
}```
2.3. Github Files – best parameters & results

https://colab.research.google.com/drive/1_XqJxRYZ588gaq5geqjzWYPPhVr67JS5e?usp=sharing#scrollTo=MfTY-ogAEnDF
3. User's experiences with the ML code and data shared

• I have a background in statistics/methodology and had very little knowledge of ML
• I was introduced to ML by the work of the project members
• With the ML code/data shared and assistance of ML team members, I was quickly able to experiment
• I share my experiences using ML to:
  · Identify misclassified products on the shared data
  · Use this imperfect dataset to simulate the integration of ML into a manual coding operation
  · Make many mistakes and learn many lessons along the way
• My experiences will continue and more may be added