**Experiments report**

*General task description*

In this test task, we have faced two classification problems. The solution has to predict two sets of classes: **google\_category\_l1** and **google\_category\_l2**.

For the input we have chosen the following sets of columns: **title**, **description**, **brand**, and  **ebay\_category**. We have dropped the **product\_id** due to a lack of informativeness and localized\_aspects\_json because it is duplicating the information from the **title** column.

Preprocessing techniques:

* Ebay\_category – categorical feature, for its preprocessing we have used the LabelEncoder from the scikit-learn package.
* Title, Brand - more or less clean textual features, we have applied the tokenization procedure using the all-MiniLM-L6-v2 model from the SentenceTransformer package.
* Description - textual feature. Before the same tokenization procedure as mentioned above, we filter out texts from all alphabetic and numerical data. To improve the context capturing process.
* **google\_category\_l1**, **google\_category\_l2** - we have applied the class remapping, so for example, 888 class will be 0. This step is necessary for the proper work of the model.

*Experiments*

*Experiment #1:*

Input data - preprocessed titles, brand and ebay\_category features.

**Metrics: Accuracy:** 86%

**Precision:**

cls\_0: 0.9047619,

cls\_1 : 0,

cls\_2: 0.82154882

cls\_3 : 0.87804878,

cls\_4 : 0.96183206,

cls\_5 : 0.78303199

cls\_6: 0.8125,

cls\_7: 0.83333333,

cls\_8: 1,

cls\_9: 0.93019197

cls\_10: 0,

cls\_11: 0.925,

cls\_12: 0.77777778,

Cls\_13: 0.8490566,

cls\_14: 0.79716981,

Cls\_15: 0.75342466,

cls\_16: 0.87586207,

cls\_17: 0,

cls\_18 0.78007519,

cls\_19: 0.88888889,

cls\_20: 0.91014169.

**Recall:**

cls\_0: 0.82608696

cls\_1: 0

cls\_2: 0.7625

cls\_3: 0.7826087

cls\_4: 0.85714286

cls\_5: 0.88801262

cls\_6: 0.72222222

cls\_7: 0.34883721

cls\_8: 0.5625

cls\_9: 0.93673111

cls\_10: 0

cls\_11: 0.62711864

cls\_12: 0.28

cls\_13: 0.87662338

cls\_14: 0.8622449

cls\_15: 0.85271318

cls\_16: 0.87788018

cls\_17: 0

cls\_18: 0.60583942

cls\_19: 0.20779221

cls\_20: 0.98546629

*Experiment #2*

Input data - preprocessed titles, **description**, brand, and ebay\_category features.

Metrics:

**Accuracy:** 87%

**Precision:**

cls\_0: 0.94736842

cls\_1: 0

cls\_2: 0.81107492

cls\_3: 0.92682927

cls\_4: 0.97674419

cls\_5: 0.79403409

cls\_6: 0.90909091

cls\_7: 0.76

cls\_8: 1

cls\_9: 0.93497364

cls\_10: 1

cls\_11: 0.90909091

cls\_12: 1

cls\_13: 0.8525641

cls\_14: 0.76712329

cls\_15: 0.75352113

cls\_16: 0.88036117

cls\_17: 0

cls\_18: 0.79423077

cls\_19: 0.7

cls\_20: 0.90888806

**Recall:**

cls\_0: 0.7826087

cls\_1: 0

cls\_2: 0.778125

cls\_3: 0.82608696

cls\_4: 0.85714286

cls\_5: 0.88170347

cls\_6: 0.55555556

cls\_7: 0.44186047

cls\_8: 0.5

cls\_9: 0.93497364

cls\_10: 0.5

cls\_11: 0.6779661

cls\_12: 0.44

cls\_13: 0.86363636

cls\_14: 0.85714286

cls\_15: 0.82945736

cls\_16: 0.89861751

cls\_17: 0

cls\_18: 0.60291971

cls\_19: 0.18181818

cls\_20: 0.98667743

*Conclusion*

As we can see, adding a new feature **description** does not bring significant results.

Because of that, we decided to change our preprocessing of the description column. The main idea was to use the LLM to get only useful information about the product in a condensed version. So tokenization would bring much more benefit to the predictive power of the model. We decided to use the **Phi family** models provided by Microsoft. **Phi-2** unfortunately does not meet the quality expectations. On the other hand, **Phi-3** provided very good results.