ЛАБА 3: Задание: 1) Выбрать набор данных (датасет), содержащий категориальные признаки и пропуски в данных. Для выполнения следующих пунктов можно использовать несколько различных наборов данных (один для обработки пропусков, другой для категориальных признаков и т.д.) 2) Для выбранного датасета (датасетов) на основе материалов лекции решить следующие задачи: обработку пропусков в данных; кодирование категориальных признаков; масштабирование данных.

In [36]:
```
#!pip install lightgbm
```

In [37]:
```
import pandas as pd
import numpy as np
import lightgbm # сожрет все сырым и построит регрессионную модель, которая покажет важные фичи
                # чтобы дальше делать лабу только на них
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.neighbors import KNeighborsRegressor
```

In [38]:
```
store = pd.read_csv('./data/googleplaystore.csv')
```

In [39]:
```
store.head()
```

Out[39]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Conter Ratin
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyon
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyon
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyon

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Conter Ratin
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyon

Пусть Rating - целевая фича Пропуски в данных есть. Это хорошо. Сделаем сразу две части задания

In [40]:
```
# Все колонки, которые не являются числами, делаем категориальными:
for column in store.select_dtypes(include = ['object']).columns.tolist():
    store[column] = store[column].astype('category')
```

In [41]:
```
lgbm_regressor = lightgbm.LGBMRegressor().fit(store.loc[:, store.columns != 'Rating'], store['Rating'])
lgbm_regressor # построили сырую и простую модель, вставив на X все кроме целевой, а на y - "Rating"
```

Out[41]:
```
LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
       importance_type='split', learning_rate=0.1, max_depth=-1,
       min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
       n_estimators=100, n_jobs=-1, num_leaves=31, objective=None,
       random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True,
       subsample=1.0, subsample_for_bin=200000, subsample_freq=0)
```

In [42]:
```
list_of_importances = list(zip(store.loc[:, store.columns != 'Rating'].columns.tolist(),
                               lgbm_regressor.feature_importances_))
list_of_importances = sorted(list_of_importances, key= lambda x: x[1], reverse= True) # список фич, отсортированных по важности
```

In [43]:
```
important_features = [x[0] for x in list_of_importances if x[1] > 20]
#important_features # оставим только важные фичи
```

In [44]:
```
important_features.extend(['Rating'])
store = store[important_features]
```

In [45]:
```
# заполнение пропусков
for column in store.select_dtypes(include = ['int64', 'float64']).columns.tolist():
    store[column] = store[column].fillna(store[column].mean())
for column in store.select_dtypes(include = ['category']).columns.tolist():
    store[column] = store[column].fillna(store[column].describe(include= ['category'])['top'])
```

In [46]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for column in store.select_dtypes(include = ['category']).columns.tolist():
    le.fit(store[column])
    store[column] = le.transform(store[column])
#кодирование категориальных признаков
store.head()
```

Out[46]:

	Current Ver	Reviews	Type	Rating
0	118	1182	1	4.1
1	1018	5923	1	3.9
2	464	5680	1	4.7
3	2765	1946	1	4.5
4	277	5923	1	4.3

In [47]:
```
store.loc[:, store.columns != 'Rating'] = store.loc[:, store.columns != 'Rating'].apply(lambda x: x/x.max(), axis=0)
# нормирование или масштабирование данных
```

In [48]:
```
store.head()
```

Out[48]:

	Current Ver	Reviews	Type	Rating
0	0.041681	0.196967	0.5	4.1
1	0.359590	0.987002	0.5	3.9
2	0.163900	0.946509	0.5	4.7
3	0.976687	0.324279	0.5	4.5
4	0.097845	0.987002	0.5	4.3