```
!pip install umap-learn update
```

```
Collecting umap-learn
  Downloading umap-learn-0.5.1.tar.gz (80 kB)
     |████████████████████████████████| 80 kB 4.1 MB/s
Collecting update
  Downloading update-0.0.1-py2.py3-none-any.whl (2.9 kB)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: scipy>=1.0 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: numba>=0.49 in /usr/local/lib/python3.7/dist-packages
Collecting pynndescent>=0.5
  Downloading pynndescent-0.5.5.tar.gz (1.1 MB)
     |████████████████████████████████| 1.1 MB 10.4 MB/s
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: llvmlite<0.35,>=0.34.0.dev0 in /usr/local/lib/python3
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages
Collecting style==1.1.0
  Downloading style-1.1.0-py2.py3-none-any.whl (6.4 kB)
Building wheels for collected packages: umap-learn, pynndescent
  Building wheel for umap-learn (setup.py) ... done
  Created wheel for umap-learn: filename=umap_learn-0.5.1-py3-none-any.whl size=76564
  Stored in directory: /root/.cache/pip/wheels/01/e7/bb/347dc0e510803d7116a13d592b10c
  Building wheel for pynndescent (setup.py) ... done
  Created wheel for pynndescent: filename=pynndescent-0.5.5-py3-none-any.whl size=526
  Stored in directory: /root/.cache/pip/wheels/af/e9/33/04db1436df0757c42fda8ea6796d7
Successfully built umap-learn pynndescent
Installing collected packages: style, pynndescent, update, umap-learn
Successfully installed pynndescent-0.5.5 style-1.1.0 umap-learn-0.5.1 update-0.0.1
```

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
import seaborn as sns
import numpy as np


train = pd.read_csv('/content/drive/MyDrive/ColabNotebooks/employees/employee_leave_train.
test = pd.read_csv('/content/drive/MyDrive/ColabNotebooks/employees/employee_leave_test.cs
```

Visualization

```
train.head()
```

Год

```
test.head()
```

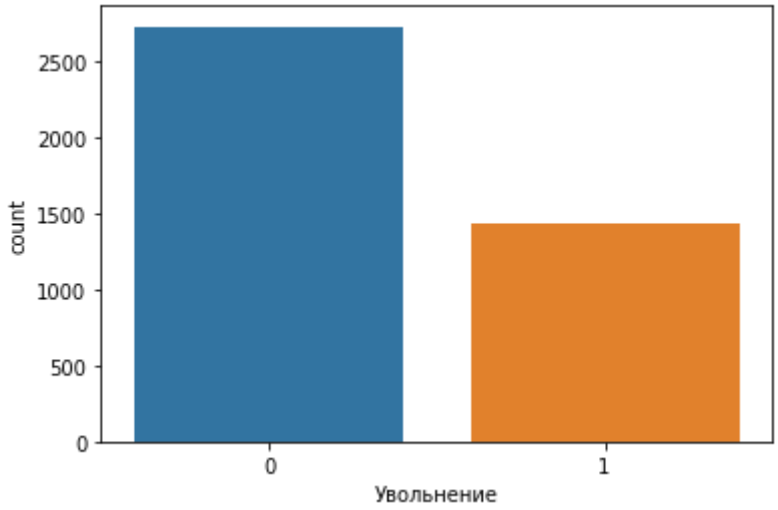
	Образование	Год начала работы	Город	Уровень оплаты	Возраст	Пол	Отстранения	Опыт
0	Masters	2017	New Delhi	2	36	Male	Yes	2
1	Masters	2015	Bangalore	3	27	Female	No	5
2	Masters	2017	New Delhi	2	33	Male	No	2
3	Bachelors	2015	Bangalore	1	25	Female	Yes	3
4	Bachelors	2015	Pune	3	24	Female	No	2

```
le_suspen = LabelEncoder()
le_suspen.fit(train['Отстранения'])
train['Отстранения'] = le_suspen.transform(train['Отстранения'])
test['Отстранения'] = le_suspen.transform(test['Отстранения'])

le_sex = LabelEncoder()
le_sex.fit(train['Пол'])
train['Пол'] = le_sex.transform(train['Пол'])
test['Пол'] = le_sex.transform(test['Пол'])

le_city = LabelEncoder()
le_city.fit(train['Город'])
train['Город'] = le_city.transform(train['Город'])
test['Город'] = le_city.transform(test['Город'])

le_ed = LabelEncoder()
le_ed.fit(train['Образование'])
train['Образование'] = le_ed.transform(train['Образование'])
test['Образование'] = le_ed.transform(test['Образование'])


sns.pairplot(train, hue='Увольнение')
```

```
<seaborn.axisgrid.PairGrid at 0x7f8fb6553490>
```

```
sns.countplot(x='Увольнение', data=train)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8fb5ab3510>
```




```
dummy_city = pd.get_dummies(train['Город'], prefix='city_')
train = pd.merge(
    left=train,
    right=dummy_city,
    left_index=True,
    right_index=True,
)

dummy_city = pd.get_dummies(test['Город'], prefix='city_')
test = pd.merge(
    left=test,
    right=dummy_city,
    left_index=True,
    right_index=True,
)


dummy_ed = pd.get_dummies(train['Образование'], prefix='ed_')
train = pd.merge(
    left=train,
    right=dummy_ed,
    left_index=True,
    right_index=True,
)

dummy_ed = pd.get_dummies(test['Образование'], prefix='ed_')
test = pd.merge(
    left=test,
    right=dummy_ed,
    left_index=True,
    right_index=True,
)


test.drop(['Образование', 'Город'], axis=1, inplace=True)
```

```
train.drop(['Образование', 'Город'], axis=1, inplace=True)

target = train['Увольнение']
train.drop(['Увольнение'], axis=1, inplace=True)
```

```
train.head()
```

	Год начала работы	Уровень оплаты	Возраст	Пол	Отстранения	Опыт	city__0	city__1	city__2	ed__0
0	2013	3	30	1	0	5	0	1	0	0
1	2018	3	25	1	0	3	1	0	0	1
2	2017	3	26	1	0	4	0	1	0	0
3	2012	1	38	1	0	5	1	0	0	0

```
test.head()
```

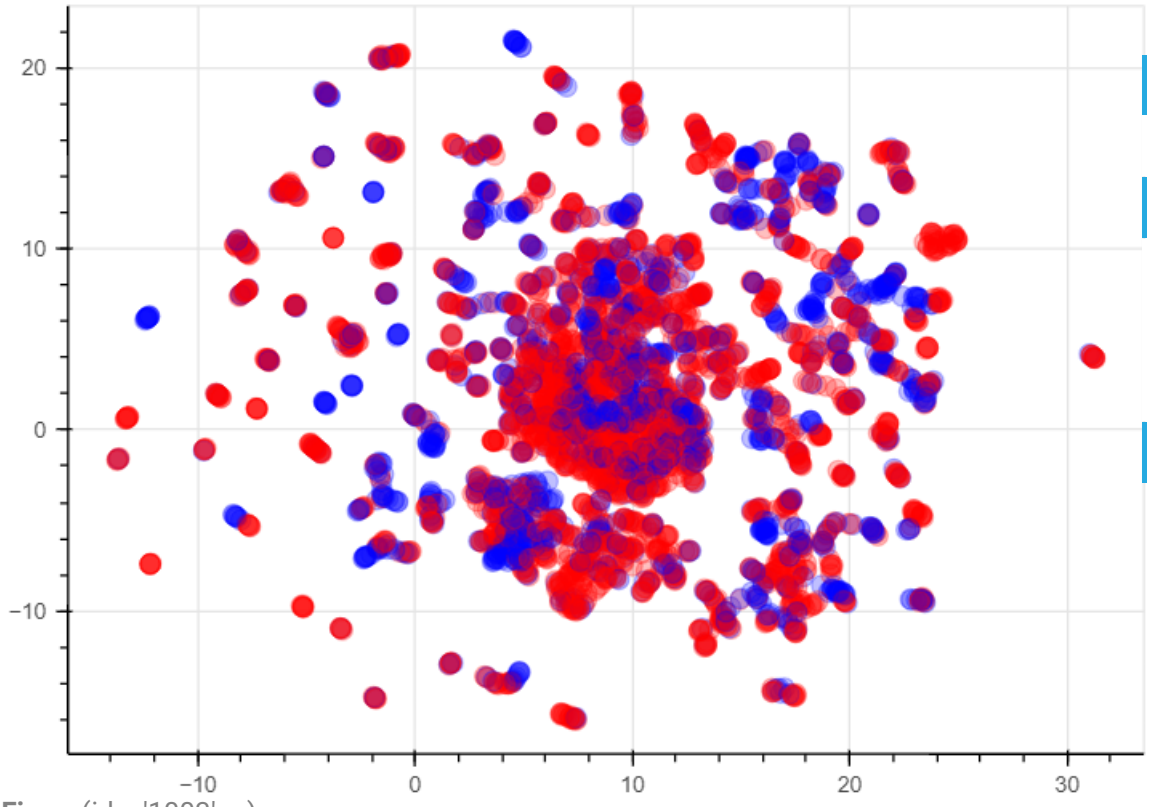
	Год начала работы	Уровень оплаты	Возраст	Пол	Отстранения	Опыт	city__0	city__1	city__2	ed__0
0	2017	2	36	1	1	2	0	1	0	0
1	2015	3	27	0	0	5	1	0	0	0
2	2017	2	33	1	0	2	0	1	0	0
3	2015	1	25	0	1	3	1	0	0	1

```
import umap

umap = umap.UMAP(n_neighbors=5, min_dist=0.5)
transformed_features = umap.fit_transform(train)
```

```
/usr/local/lib/python3.7/dist-packages/numba/np/ufunc/parallel.py:363: NumbaWarning:
  warnings.warn(problem)
```

```
import bokeh.models as bm, bokeh.plotting as pl
from bokeh.io import output_notebook
output_notebook()

def draw_vectors(x, y, radius=10, alpha=0.25, color='blue',
                 width=600, height=400, show=True, **kwargs):
    """ draws an interactive plot for data points with auxilirary info on hover """
    data_source = bm.ColumnDataSource({ 'x' : x, 'y' : y, 'color': color, **kwargs })

    fig = pl.figure(active_scroll='wheel_zoom', width=width, height=height)
    fig.scatter('x', 'y', size=radius, color='color', alpha=alpha, source=data_source)

    fig.add_tools(bm.HoverTool(tooltips=[(key, "@" + key) for key in kwargs.keys()]))
```

```
    if show: pl.show(fig)
    return fig


draw_vectors(
    transformed_features[:, 0],
    transformed_features[:, 1],
    color=[["red", "blue"][t] for t in target]
    )
```



Figure(id = '1003', ...)

LogisticRegression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score


log = LogisticRegression()
log.fit(train, target)
pred = log.predict(train)
f1_score(target, pred)
```

```
    0.5193661971830986
```

```
pred = log.predict(test)
submission = pd.DataFrame(pred)
submission.head()
```

	0
0	0
1	1
2	0
3	1

```
submission.to_csv('logistic.csv', index=False, header=False)
```

RandomForest

```
from sklearn.ensemble import RandomForestClassifier
```

```
clf = RandomForestClassifier(random_state=5, class_weight='balanced', n_estimators=150)
clf.fit(train, target)
pred = clf.predict(train)
f1_score(target, pred)
```

```
0.8913352272727272
```

```
pred = clf.predict(test)
submission = pd.DataFrame(pred)
submission.head()
```

	0
0	0
1	1
2	0
3	0
4	1

```
submission.to_csv('randomforest_improved.csv', index=False, header=False)
```

CatBoost

```
!pip install catboost
```

```
from catboost import CatBoostClassifier, Pool, cv
from sklearn.utils.class_weight import compute_class_weight
import numpy as np
```

```
classes = np.unique(target)
weights = compute_class_weight(class_weight='balanced', classes=classes, y=target)
class_weights = dict(zip(classes, weights))
```

```
cat = CatBoostClassifier()

grid = {'learning_rate': [0.03, 0.1, 0.08],
        'depth': [4, 6],
        'l2_leaf_reg': [1, 3, 5, 7]}

grid_search_result = cat.grid_search(grid, X=train, y=target)
```

```
grid_search_result['params']
```

```
{'depth': 4, 'l2_leaf_reg': 3, 'learning_rate': 0.1}
```

```
cat = CatBoostClassifier(**grid_search_result['params'], class_weights=class_weights, rand

cat.fit(train,target,verbose=False)
pred = cat.predict(train)
f1_score(target, pred)
```

```
0.8416092787241756
```

```
pred = cat.predict(test)
submission = pd.DataFrame(pred)
submission.head()
```

	0
0	0
1	1
2	0
3	1
4	1

```
submission.to_csv('cat_boost.csv', index=False, header=False)
```

```
pred = kde.predict(test)
submission = pd.DataFrame(pred)
submission.head()
```

```
submission.to_csv('kde.csv', index=False, header=False)
```

✓ 0 сек. выполнено в 14:26