



Министерство образования и науки Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)

КАФЕДРА «Системы обработки информации и управления» (ИУ5)

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  
***К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ***  
***НА ТЕМУ:***

**«Использование алгоритма CatBoost для выявления факторов, ведущих к  
увольнению сотрудников»**

Студент группы ИУ5-32М \_\_\_\_\_ Демьянчук Г.В.

Руководитель \_\_\_\_\_ Гапанюк Ю.Е.

2020 г.

## Содержание

Введение .....	3
Исследование данных .....	4
Предварительная обработка данных .....	8
Классификация .....	13
Обучение .....	13
Вывод.....	20
Список использованной литературы.....	21

## **Введение**

Для работодателей очень важно анализировать причины увольнения своих сотрудников. В данной курсовой работе будет использован датасет, позволяющий выявить основные факторы, приводящие сотрудника к неудовлетворённости своей работой. Будет проведена подготовка данных и их анализ. С использованием библиотеки CatBoost будут выявлены факторы ведущие к увольнению сотрудников.

## Исследование данных

### Установка пакета Yandex CatBoost

```
!pip install catboost
```

### Импорт необходимых пакетов: Numpy, Pandas, Matplotlib, Seaborn, Scikit-learn и CatBoost

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_selection import VarianceThreshold
```

Загружаем датасет. Это вымышленный набор данных, созданный специалистами по данным IBM. <https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset>

```
ibm_hr_df = pd.read_csv("/content/sample_data/IBM-HR-Employee-Attrition.csv")
ibm_hr_df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipSatisfaction	St.
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	1	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	4	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	3	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	4	

5 rows x 35 columns

Рис.1 – Просмотр заголовка

Просмотр типов данных. Это данные типа object, int.

```
ibm_hr_df.info()
```

```

↳ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
Age                1470 non-null int64
Attrition          1470 non-null object
BusinessTravel     1470 non-null object
DailyRate         1470 non-null int64
Department        1470 non-null object
DistanceFromHome  1470 non-null int64
Education         1470 non-null int64
EducationField     1470 non-null object
EmployeeCount     1470 non-null int64
EmployeeNumber    1470 non-null int64
EnvironmentSatisfaction 1470 non-null int64
Gender            1470 non-null object
HourlyRate        1470 non-null int64
JobInvolvement    1470 non-null int64
JobLevel         1470 non-null int64
JobRole          1470 non-null object
JobSatisfaction   1470 non-null int64
MaritalStatus     1470 non-null object
MonthlyIncome     1470 non-null int64
MonthlyRate       1470 non-null int64
NumCompaniesWorked 1470 non-null int64
Over18           1470 non-null object
OverTime          1470 non-null object
PercentSalaryHike 1470 non-null int64
PerformanceRating 1470 non-null int64
RelationshipSatisfaction 1470 non-null int64
StandardHours     1470 non-null int64
StockOptionLevel  1470 non-null int64
TotalWorkingYears 1470 non-null int64
TrainingTimesLastYear 1470 non-null int64
WorkLifeBalance   1470 non-null int64
YearsAtCompany    1470 non-null int64
YearsInCurrentRole 1470 non-null int64
YearsSinceLastPromotion 1470 non-null int64
YearsWithCurrManager 1470 non-null int64
dtypes: int64(26), object(9)
memory usage: 402.0+ KB

```

Рис.2 – Типы данных

## Получение сводной статистики набора данных IBM HR

```
ibm_hr_df.describe()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	...	Reli
<b>count</b>	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	...	...
<b>mean</b>	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156	2.729932	2.063946	...	...
<b>std</b>	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428	0.711561	1.106940	...	...
<b>min</b>	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000	1.000000	1.000000	...	...
<b>25%</b>	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000	2.000000	1.000000	...	...
<b>50%</b>	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000	3.000000	2.000000	...	...
<b>75%</b>	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000	3.000000	3.000000	...	...
<b>max</b>	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000	4.000000	5.000000	...	...

8 rows x 26 columns

Рис.3 – Сводная статистика

## Выявление нерелевантных атрибутов EmployeeCount и StandardHours

```
irrList = ['EmployeeCount', 'StandardHours']
ibm_hr_df[irrList].describe()
```

	EmployeeCount	StandardHours
<b>count</b>	1470.0	1470.0
<b>mean</b>	1.0	80.0
<b>std</b>	0.0	0.0
<b>min</b>	1.0	80.0
<b>25%</b>	1.0	80.0
<b>50%</b>	1.0	80.0
<b>75%</b>	1.0	80.0
<b>max</b>	1.0	80.0

Рис.4 - Атрибуты EmployeeCount и StandardHours

## Выявление нерелевантного атрибута Over18

```
ibm_hr_df["Over18"].value_counts()
```

```
Y    1470
Name: Over18, dtype: int64
```

Рис.5 – Атрибут Over18

Из сводной статистики видно, что атрибуты EmployeeCount, StandardHours и Over18 содержат только одно значение для всех 1470 записей

- EmployeeCount содержит только одно значение - 1.0
- StandardHours содержит только одно значение - 80.0
- Over18 содержит только одно значение - 'Y'

Эти нерелевантные атрибуты удалим из набора данных

Проверка на неопределённые и отсутствующие значения

```
ibm_hr_df.isnull().sum(axis=0)
```

Age	0
Attrition	0
BusinessTravel	0
DailyRate	0
Department	0
DistanceFromHome	0
Education	0
EducationField	0
EmployeeCount	0
EmployeeNumber	0
EnvironmentSatisfaction	0
Gender	0
HourlyRate	0
JobInvolvement	0
JobLevel	0
JobRole	0
JobSatisfaction	0
MaritalStatus	0
MonthlyIncome	0
MonthlyRate	0
NumCompaniesWorked	0
Over18	0
Overtime	0
PercentSalaryHike	0
PerformanceRating	0
RelationshipSatisfaction	0
StandardHours	0
StockOptionLevel	0
TotalWorkingYears	0
TrainingTimesLastYear	0
WorkLifeBalance	0
YearsAtCompany	0
YearsInCurrentRole	0
YearsSinceLastPromotion	0
YearsWithCurrManager	0
dtype: int64	

Рис.6 - Проверка на неопределённые и отсутствующие значения

Что ж, нам здесь повезло, в этом наборе данных нет пропущенных значений.

Далее проверим наличие повторяющихся записей в наборе данных.

```
ibm_hr_df.duplicated().sum()
```

В наборе данных также нет повторяющихся записей.

Преобразование двоичного категориального атрибута OverTime в {1, 0}

```
ibm_hr_df['OverTime'].replace(to_replace=dict(Yes=1, No=0), inplace=True)
```

## **Предварительная обработка данных**

Удаление нерелевантных атрибутов

```
ibm_hr_df = ibm_hr_df.drop(['EmployeeCount', 'StandardHours', 'Over18'], axis=1)
```

Выполнение корреляционного анализа Пирсона между атрибутами для облегчения уменьшения размерности

```
plt.figure(figsize=(16, 16))
sns.heatmap(ibm_hr_df.corr(), annot=True, fmt=".2f")

plt.show()
```



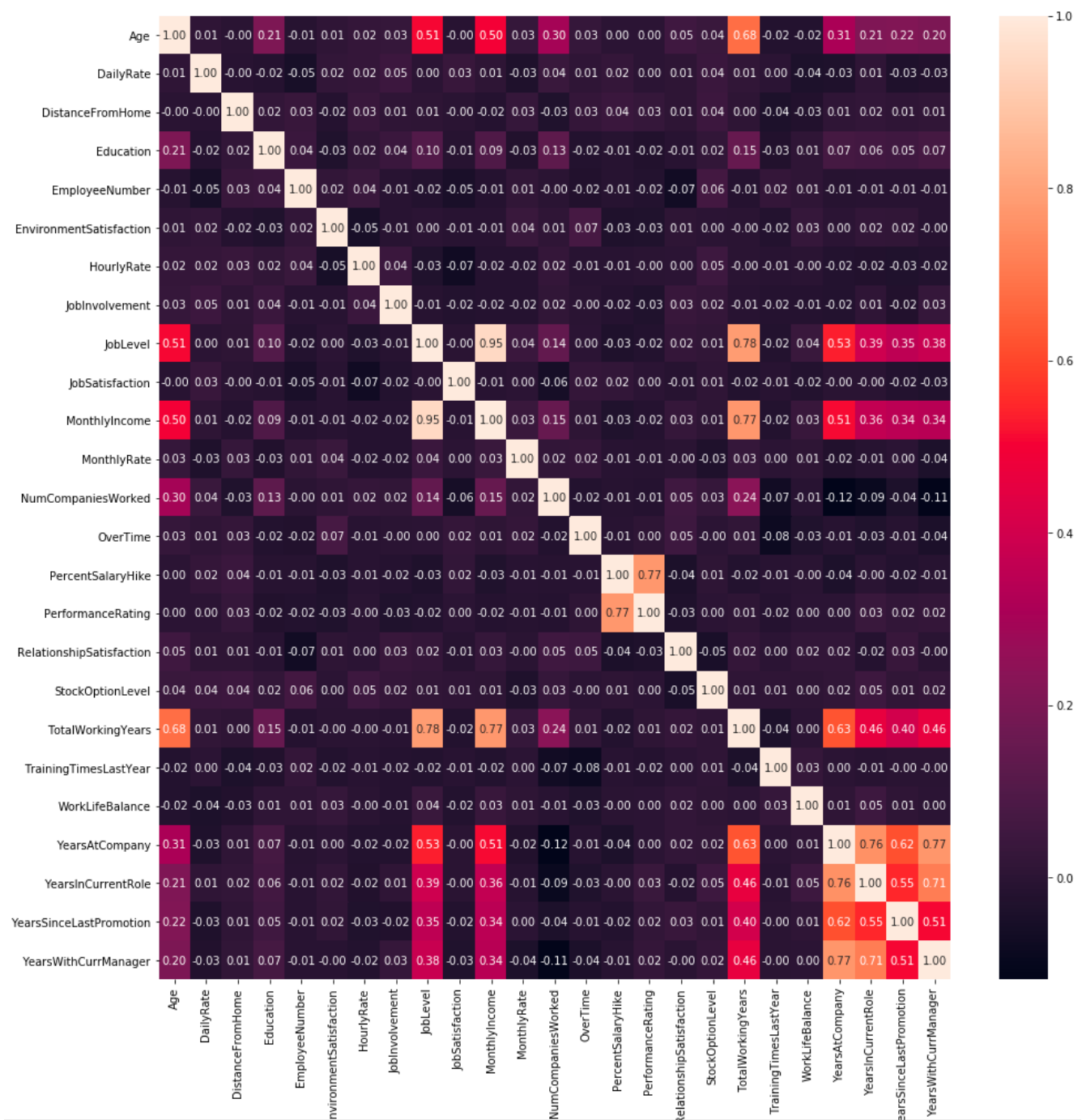


Рис.7 – Корреляционный анализ

## Выполнение дисперсионного анализа

```
variance_x = ibm_hr_df.drop('Attrition', axis=1)
variance_one_hot = pd.get_dummies(variance_x)
```

#Нормализовать набор данных. Это необходимо для получения порога дисперсии.

```

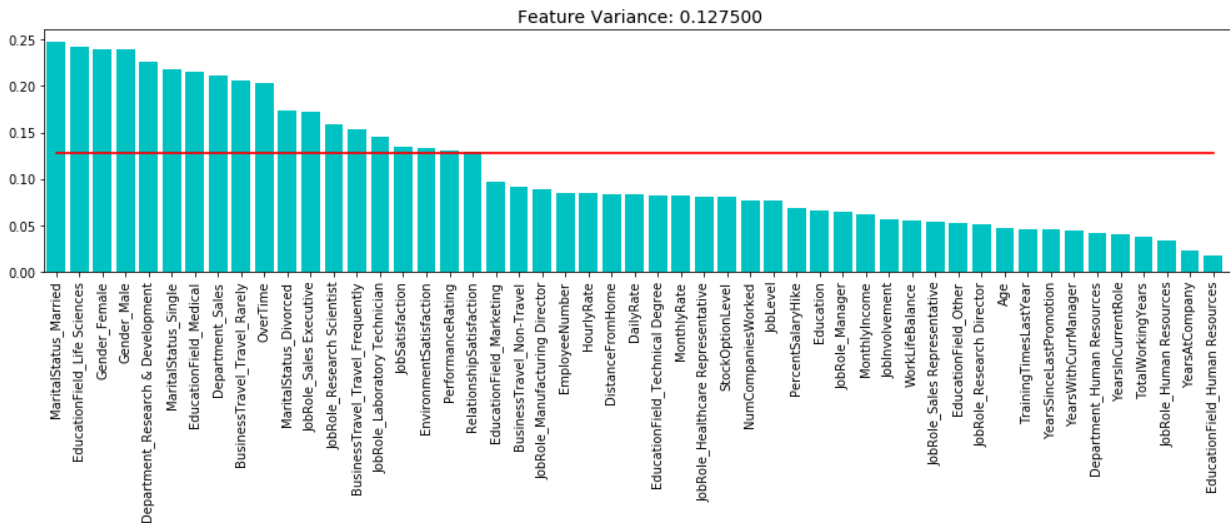
scaler = MinMaxScaler()
scaler.fit(variance_one_hot)
MinMaxScaler(copy=True, feature_range=(0, 1))
scaled_variance_one_hot = scaler.transform(variance_one_hot)

#Установить пороговые значения и запустить VarianceThreshold
thres = .85* (1 - .85)
sel = VarianceThreshold(threshold=thres)
sel.fit(scaled_variance_one_hot)
variance = sel.variances_

#Сортировка в порядке возрастания для построения графика
indices = np.argsort(variance)[:, -1]
feature_list = list(variance_one_hot)
sorted_feature_list = []
thres_list = []
for f in range(len(variance_one_hot.columns)):
    sorted_feature_list.append(feature_list[indices[f]])
    thres_list.append(thres)

plt.figure(figsize=(14, 6))
plt.title("Feature Variance: %f" %(thres), fontsize = 14)
plt.bar(range(len(variance_one_hot.columns)), variance[indices], color="c")
plt.xticks(range(len(variance_one_hot.columns)), sorted_feature_list, rotation
= 90)
plt.xlim([-0.5, len(variance_one_hot.columns)])
plt.plot(range(len(variance_one_hot.columns)), thres_list, "k-", color="r")
plt.tight_layout()
plt.show()

```



```
rAttrList = ['Department', 'OverTime', 'HourlyRate',
             'StockOptionLevel', 'DistanceFromHome',
             'YearsInCurrentRole', 'Age']
```

```
#храните только список атрибутов в rAttrList
label_hr_df = ibm_hr_df[rAttrList]
```

```
#преобразование непрерывного атрибута DistanceFromHome в категориальный
#: 1: близко, 2: среднее расстояние, 3: далеко
```

```
maxValues = label_hr_df['DistanceFromHome'].max()
minValues = label_hr_df['DistanceFromHome'].min()
intervals = (maxValues - minValues)/3
bins = [0, (minValues + intervals), (maxValues - intervals), maxValues]
groupName = [1, 2, 3]
label_hr_df['CatDistanceFromHome'] = pd.cut(label_hr_df['DistanceFromHome'], bins, labels = groupName)
```

```
# приведение типа к int64
```

```
label_hr_df['CatDistanceFromHome'] = pd.to_numeric(label_hr_df['CatDistanceFromHome'])
label_hr_df.drop(['DistanceFromHome'], axis = 1, inplace = True)
```

```
#переместить названия подразделений в 0 & 1, 0: R&D, and 1: Non-R&D
```

```
label_hr_df['Department'].replace(['Research & Development', 'Human Resources',
                                   'Sales'],
                                   [0, 1, 1], inplace = True)
```

```
#нормализация данных
```

```

label_hr_df_norm = (label_hr_df - label_hr_df.min()) / (label_hr_df.max() - label_hr_df.min())

#создать data frame для значения функций и меток классов
value_df = pd.DataFrame(columns = ['ClassValue'])

#вычислить значение класса
for row in range (0, ibm_hr_df.shape[0]):
    if label_hr_df_norm['Department'][row] == 0:
        value = 0.3 * label_hr_df_norm['HourlyRate'][row] - 0.2 * label_hr_df_norm['OverTime'][row] + \
            - 0.2 * label_hr_df_norm['CatDistanceFromHome'][row] + 0.15 * label_hr_df_norm['StockOptionLevel'][row] + \
            0.1 * label_hr_df_norm['Age'][row] - 0.05 * label_hr_df_norm['YearsInCurrentRole'][row]

    else:
        value = 0.2 * label_hr_df_norm['HourlyRate'][row] - 0.3 * label_hr_df_norm['OverTime'][row] + \
            - 0.15 * label_hr_df_norm['CatDistanceFromHome'][row] + 0.2 * label_hr_df_norm['StockOptionLevel'][row] + \
            0.05 * label_hr_df_norm['Age'][row] - 0.1 * label_hr_df_norm['YearsInCurrentRole'][row]
        value_df.loc[row] = value

# top 500 высшего класса довольны своей работой
v1 = value_df.sort_values('ClassValue', ascending = False).reset_index(drop = True)\
    ['ClassValue'][499]
# следующие top 500 нейтральны
v2 = value_df.sort_values('ClassValue', ascending = False).reset_index(drop = True)\
    ['ClassValue'][999]
# остальные недовольны своей работой

label_df = pd.DataFrame(columns = ['ClassLabel'])

#вычислить classlabel
for row in range (0, value_df.shape[0]):
    if value_df['ClassValue'][row] >= v1:
        cat = "Satisfied"
    elif value_df['ClassValue'][row] >= v2:
        cat = "Neutral"

```

```

else:
    cat = "Unsatisfied"
    label_df.loc[row] = cat

```

```
df = pd.concat([libm_hr_df, label_df], axis = 1)
```

## Классификация

```

df = df[['Age', 'Department', 'DistanceFromHome', 'HourlyRate', 'OverTime', 'StockOptionLevel',
        'MaritalStatus', 'YearsInCurrentRole', 'EmployeeNumber', 'ClassLabel']]

```

Разбиение данных на attributes/features X и label/class y

```

X = df.drop('ClassLabel', axis=1)
y = df.ClassLabel

```

Замена label/class значений из 'Satisfied', 'Neutral' и 'Unsatisfied' в 2, 1 and 0

```
y.replace(to_replace=dict(Satisfied=2, Neutral=1, Unsatisfied=0), inplace=True)
```

Выполнение 'one hot encoding' метода

```
one_hot = pd.get_dummies(X)
```

Нормализация функции

```

one_hot = (one_hot - one_hot.mean()) / (one_hot.max() - one_hot.min())
categorical_features_indices = np.where(one_hot.dtypes != np.float)[0]

```

## Обучение

Теперь давайте разделим наши данные на обучающий (70%) и тестовый (30%) набор:

```
from sklearn.model_selection import train_test_split
```

```

X_train, X_test, y_train, y_test = train_test_split(one_hot, y, train_size=0.7,
    random_state=1234)

```

```

model = CatBoostClassifier(
    custom_loss = ['Accuracy'],
    random_seed = 100,
    loss_function = 'MultiClass'
)

model.fit(
    X_train, y_train,
    cat_features = categorical_features_indices,
    verbose = True,
    #plot = True
)

    Learning rate set to 0.079242
    0:      learn: 1.0312448      total: 3.53ms      remaining: 3.52s
    1:      learn: 0.9667097      total: 6.14ms      remaining: 3.06s
    2:      learn: 0.9189766      total: 8.72ms      remaining: 2.9s
    3:      learn: 0.8786445      total: 11.2ms      remaining: 2.79s
    4:      learn: 0.8369363      total: 13.8ms      remaining: 2.75s
    5:      learn: 0.7954649      total: 16.3ms      remaining: 2.7s
    6:      learn: 0.7634820      total: 18.8ms      remaining: 2.67s
    7:      learn: 0.7342294      total: 21.8ms      remaining: 2.7s
    8:      learn: 0.7094003      total: 24.4ms      remaining: 2.68s
    9:      learn: 0.6845187      total: 27.3ms      remaining: 2.7s
    10:     learn: 0.6590169      total: 30.4ms      remaining: 2.74s
    11:     learn: 0.6395683      total: 33.2ms      remaining: 2.73s

    994:     learn: 0.0227453      total: 2.68s      remaining: 13.5ms
    995:     learn: 0.0227280      total: 2.68s      remaining: 10.8ms
    996:     learn: 0.0227062      total: 2.69s      remaining: 8.08ms
    997:     learn: 0.0226718      total: 2.69s      remaining: 5.39ms
    998:     learn: 0.0226477      total: 2.69s      remaining: 2.69ms
    999:     learn: 0.0226263      total: 2.69s      remaining: 0us
    <catboost.core.CatBoostClassifier at 0x7fbc7de450>

```

Рис.8 – Обучение

```

feature_score = pd.DataFrame(list(zip(one_hot.dtypes.index, model.get_feature_i
mportance(Pool(one_hot, label=y, cat_features=categorical_features_indices)))),
    columns=['Feature', 'Score'])
feature_score = feature_score.sort_values(by='Score', ascending=False, inplace=
False, kind='quicksort', na_position='last')

plt.rcParams["figure.figsize"] = (12,7)
ax = feature_score.plot('Feature', 'Score', kind='bar', color='c')
ax.set_title("Catboost Feature Importance Ranking", fontsize = 14)
ax.set_xlabel('')

```

```

rects = ax.patches

# get feature score as labels round to 2 decimal
labels = feature_score['Score'].round(2)

for rect, label in zip(rects, labels):
    height = rect.get_height()
    ax.text(rect.get_x() + rect.get_width()/2, height + 0.35, label, ha='center', va='bottom')

plt.show()

```

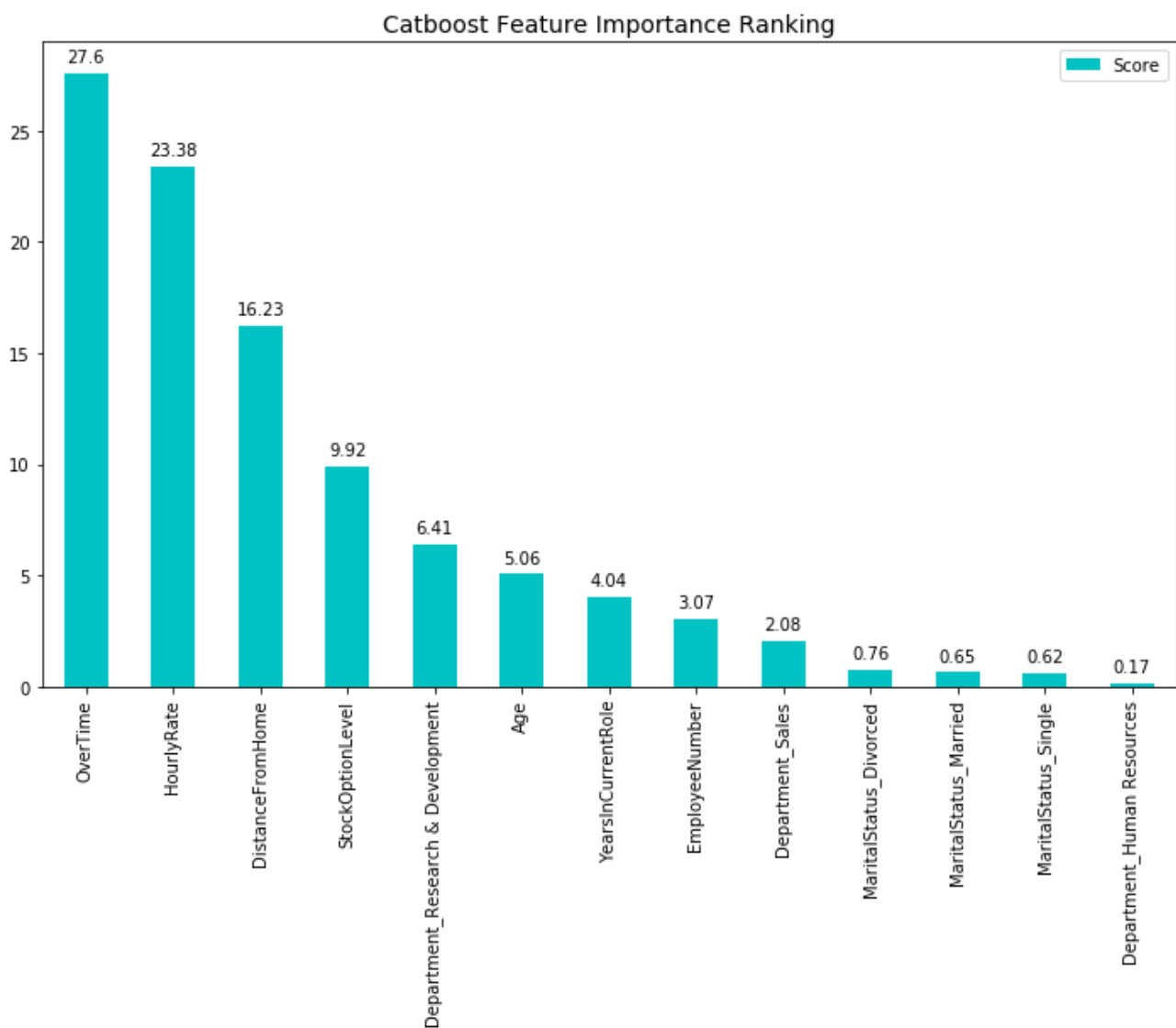


Рис.9 – Важность признаков

Как видно по графику, что наибольшей важностью обладает признак OverTime.

```
model.score(X_test, y_test)
0.9251700680272109
```

## CatBoost настройка классификатора

```
model = CatBoostClassifier(
    l2_leaf_reg = 3,
    iterations = 1000,
    fold_len_multiplier = 1.05,
    learning_rate = 0.05,
    custom_loss = ['Accuracy'],
    random_seed = 100,
    loss_function = 'MultiClass'
)

model.fit(
    X_train, y_train,
    cat_features = categorical_features_indices,
    verbose = True,
    #plot = True
)
```

0:	learn: 1.0555929	total: 2.82ms	remaining: 2.81s
1:	learn: 1.0125553	total: 5.68ms	remaining: 2.83s
2:	learn: 0.9792847	total: 8.16ms	remaining: 2.71s
3:	learn: 0.9478631	total: 10.6ms	remaining: 2.64s
4:	learn: 0.9167174	total: 13.4ms	remaining: 2.66s
5:	learn: 0.8869031	total: 15.8ms	remaining: 2.62s
994:	learn: 0.0397018	total: 2.74s	remaining: 13.8ms
995:	learn: 0.0396683	total: 2.74s	remaining: 11ms
996:	learn: 0.0396387	total: 2.74s	remaining: 8.25ms
997:	learn: 0.0395917	total: 2.74s	remaining: 5.5ms
998:	learn: 0.0395204	total: 2.75s	remaining: 2.75ms
999:	learn: 0.0394835	total: 2.75s	remaining: 0us

<catboost.core.CatBoostClassifier at 0x7fbcdc8c4d90>

Рис.10 – Обучение

```
feature_score = pd.DataFrame(list(zip(one_hot.dtypes.index, model.get_feature_i
mportance(Pool(one_hot, label=y, cat_features=categorical_features_indices)))),
    columns=['Feature', 'Score'])
```



```

feature_score = feature_score.sort_values(by='Score', ascending=False, inplace=
False, kind='quicksort', na_position='last')

plt.rcParams["figure.figsize"] = (12,7)
ax = feature_score.plot('Feature', 'Score', kind='bar', color='c')
ax.set_title("Catboost Feature Importance Ranking", fontsize = 14)
ax.set_xlabel('')

rects = ax.patches

# get feature score as labels round to 2 decimal
labels = feature_score['Score'].round(2)

for rect, label in zip(rects, labels):
    height = rect.get_height()
    ax.text(rect.get_x() + rect.get_width()/2, height + 0.35, label, ha='center
', va='bottom')

plt.show()
#plt.savefig("image.png")

```

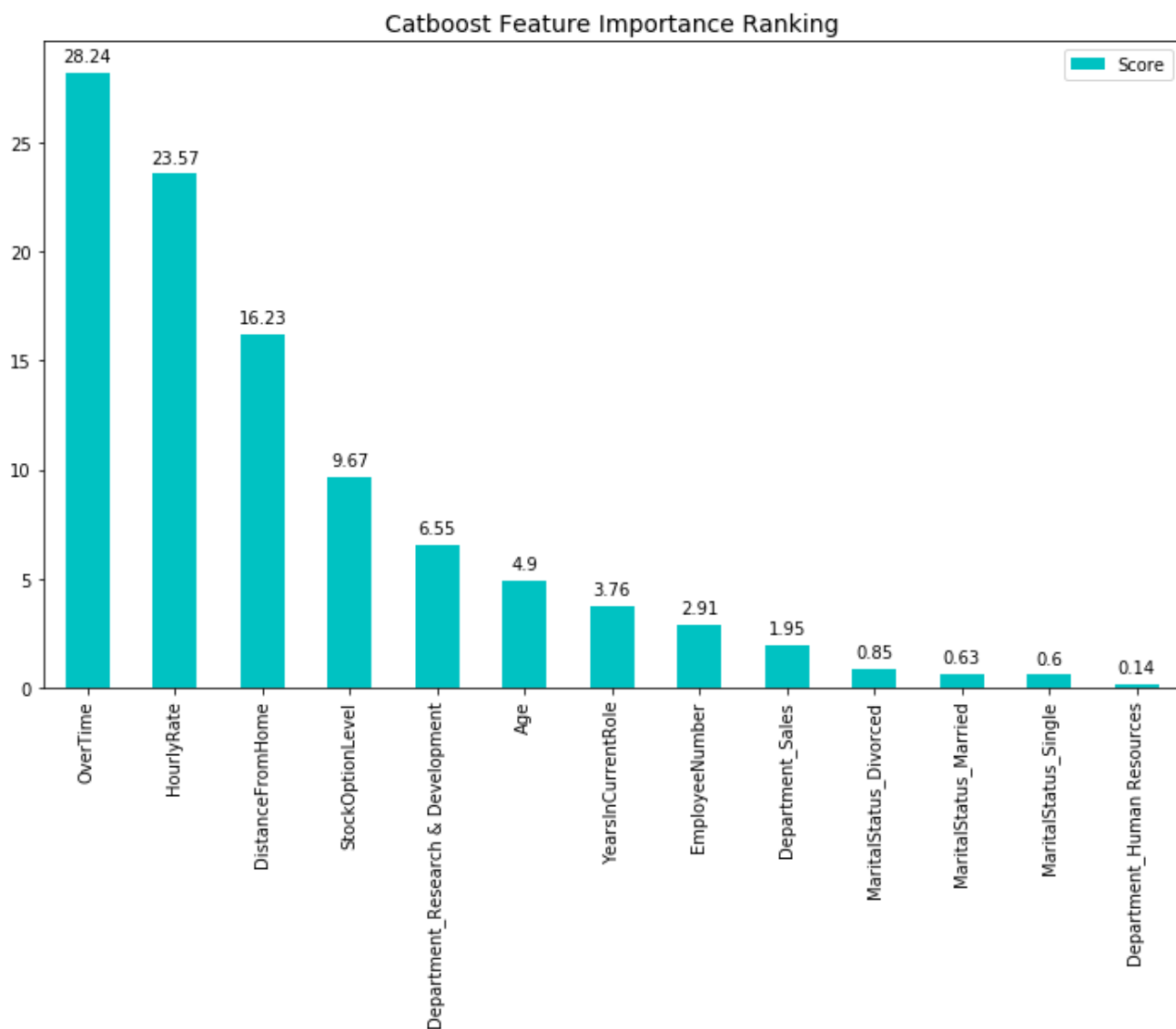


Рис.11 – Важность признаков

Как видно по графику, что наибольшей важностью обладает признак OverTime.

```
cm = pd.DataFrame()
cm['Satisfaction'] = y_test
cm['Predict'] = model.predict(X_test)
```

```
mappingSatisfaction = {0: 'Unsatisfied', 1: 'Neutral', 2: 'Satisfied'}
mappingPredict = {0.0: 'Unsatisfied', 1.0: 'Neutral', 2.0: 'Satisfied'}
cm = cm.replace({'Satisfaction': mappingSatisfaction, 'Predict': mappingPredict
})
```

```
pd.crosstab(cm['Satisfaction'], cm['Predict'], margins=True)
```

	Predict Neutral	Satisfied	Unsatisfied	All
Satisfaction				
Neutral	144	11	5	160
Satisfied	9	136	0	145
Unsatisfied	8	0	128	136
All	161	147	133	441

Рис.12 – Таблица удовлетворённости сотрудников

```
model.score(X_test, y_test)
0.9251700680272109
```

## **Вывод**

В данной курсовой работе была произведена подготовка и анализ данных факторов, влияющих на увольнение сотрудников. С использованием библиотеки CatBoost были выявлены наиболее значимые факторы.

### Список использованной литературы

1. <https://catboost.ai/docs> - CatBoost documentation
2. <https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset> - kaggle
3. <https://gist.github.com/talperetz/6030f4e9997c249b09409dcf00e78f91> - Catboost Playground
4. <https://towardsdatascience.com/https-medium-com-talperetz24-mastering-the-new-generation-of-gradient-boosting-db04062a7ea2> - Mastering The New Generation of Gradient Boosting - Tal Peretz