|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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

Введение3

Исследование данных4

Предварительная обработка данных8

Классификация13

Обучение13

Вывод20

Список использованной литературы21

**Введение**

Для работодателей очень важно анализировать причины увольнения своих сотрудников. В данной куровой 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()

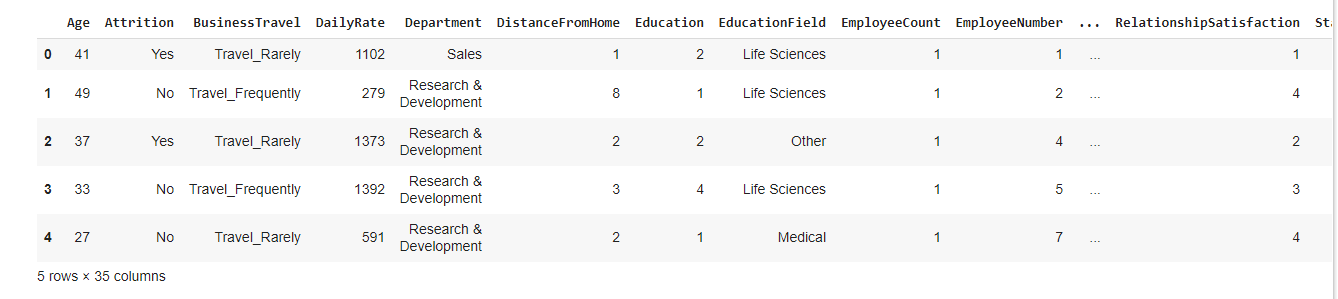


Рис.1 – Просмотр заголовка

Просмотр типов данных. Это данные типа object, int.

ibm\_hr\_df.info()



Рис.2 – Типы данных

Получение сводной статистики набора данных IBM HR

ibm\_hr\_df.describe()

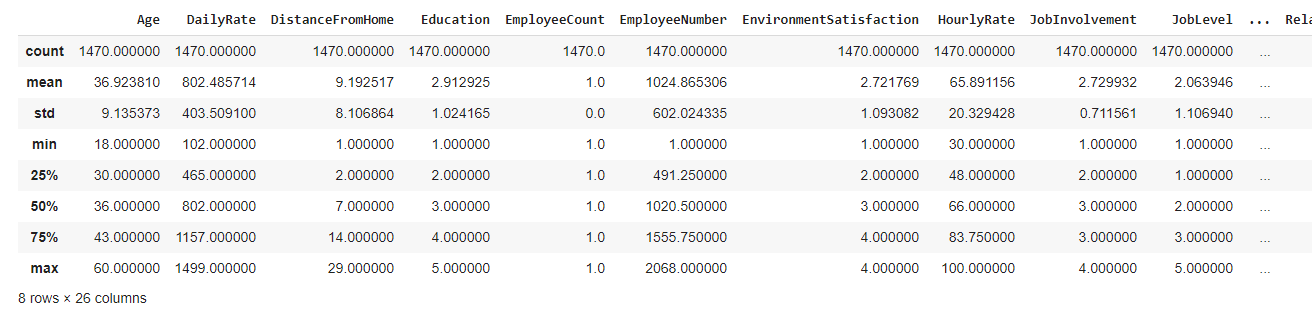


Рис.3 – Сводная статистика

Выявление нерелевантных атрибутов EmployeeCount и StandardHours

irrList = ['EmployeeCount', 'StandardHours']

ibm\_hr\_df[irrList].describe()

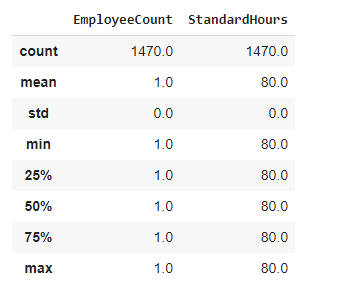


Рис.4 - Атрибуты EmployeeCount и StandardHours

Выявление нерелевантного атрибута Over18

ibm\_hr\_df["Over18"].value\_counts()



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)

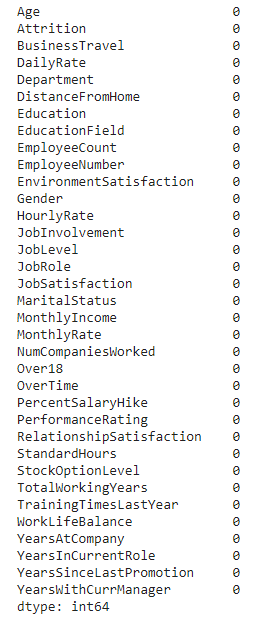


Рис.6 - Проверка на неопределённые и отсутствующие значения

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()

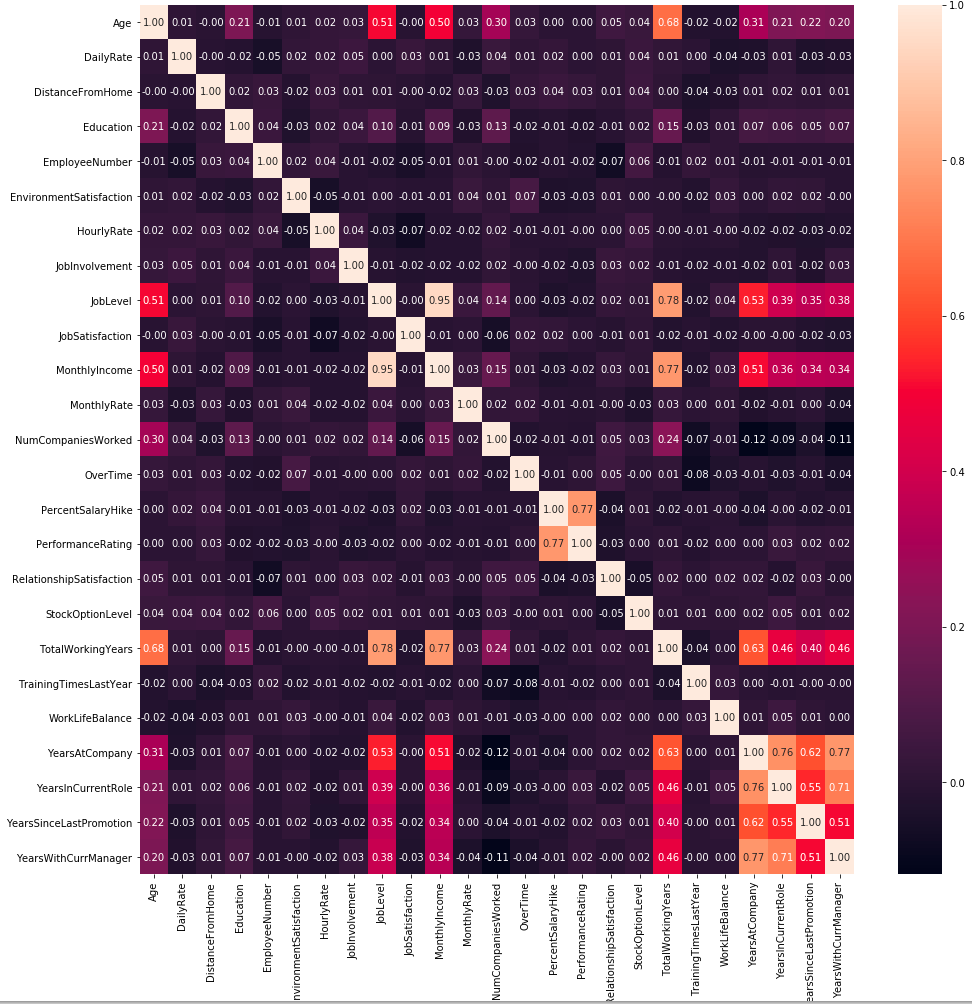


Рис.7 – Корреляционный анализ

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")

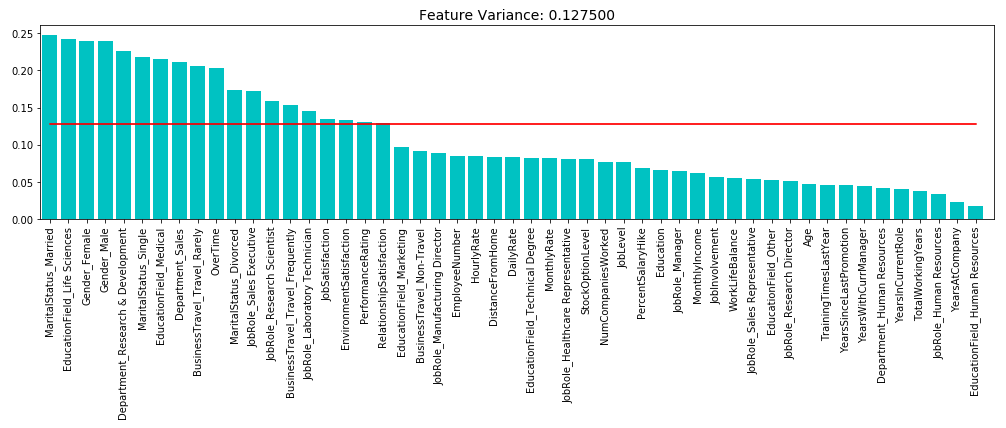
plt.xticks(range(len(variance\_one\_hot.columns)), sorted\_feature\_list, rotation = 90)

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 растояние, 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 неудовлетворены 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([ibm\_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(

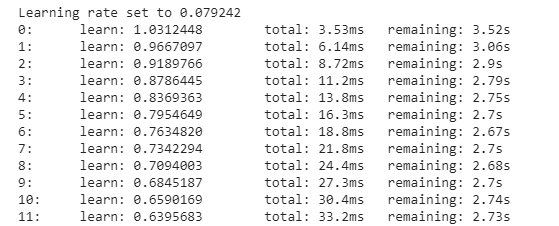
    X\_train, y\_train,

    cat\_features = categorical\_features\_indices,

    verbose = True,

    #plot = True

)



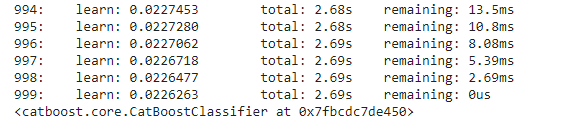


Рис.8 – Обучение

feature\_score = pd.DataFrame(list(zip(one\_hot.dtypes.index, model.get\_feature\_importance(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()

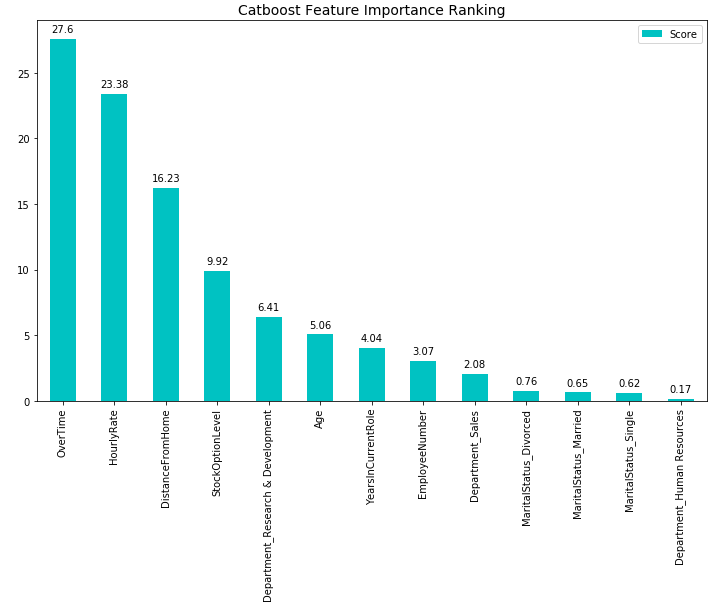


Рис.9 – Важность признаков

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(

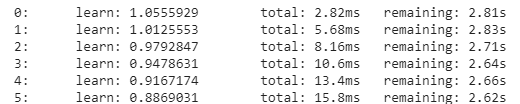
    X\_train, y\_train,

    cat\_features = categorical\_features\_indices,

    verbose = True,

    #plot = True

)



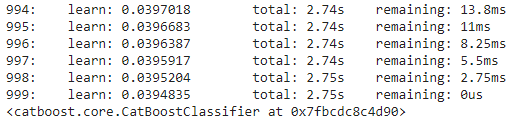


Рис.10 – Обучение

feature\_score = pd.DataFrame(list(zip(one\_hot.dtypes.index, model.get\_feature\_importance(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")

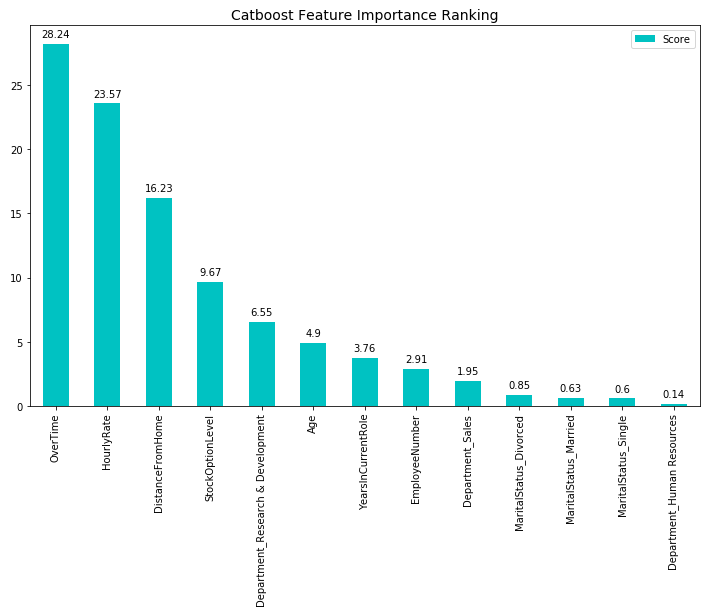


Рис.11 – Важность признаков

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)

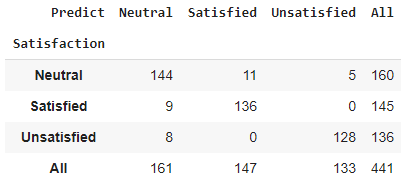


Рис.12 – Таблица удовлетворённости сотрудников

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 использование 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