**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Экзаменационное задание**

**по дисциплине: «Машинное обучение и анализ данных»**

**Работу выполнил**

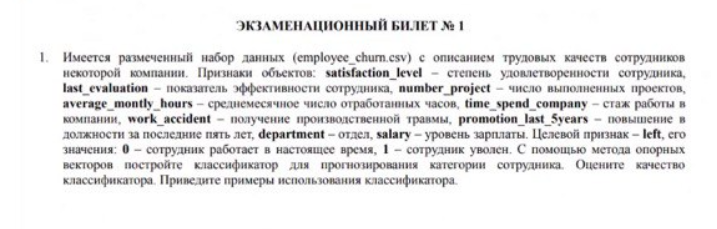
**Самсонов А.Д.**

**студент группы 201-331**

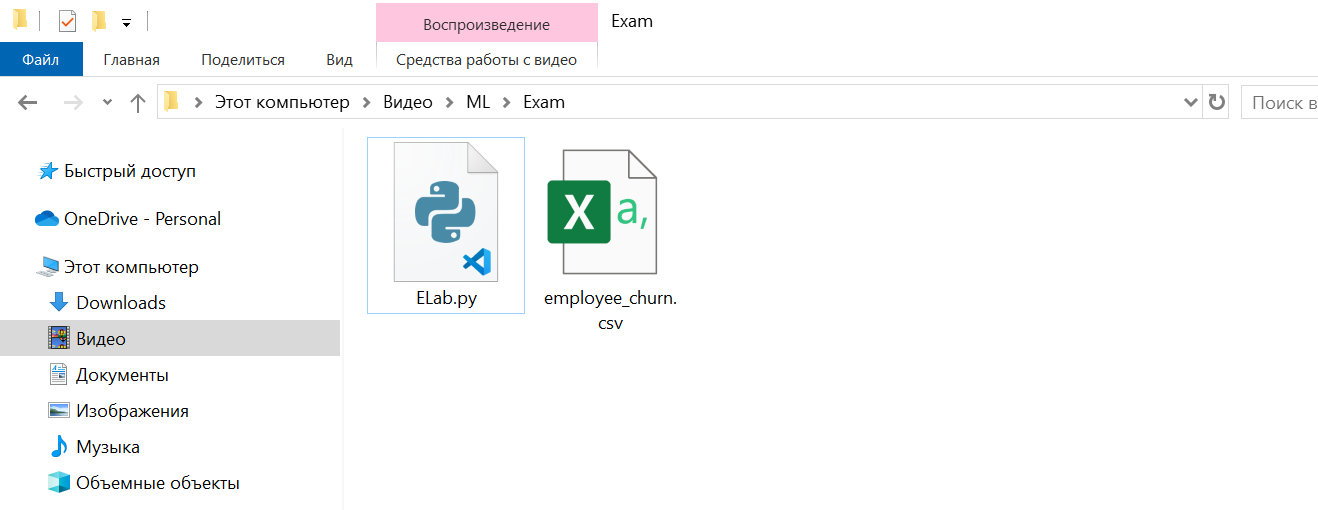
**Проверила: Харченко Е.А.**

**Москва 2024 г.**

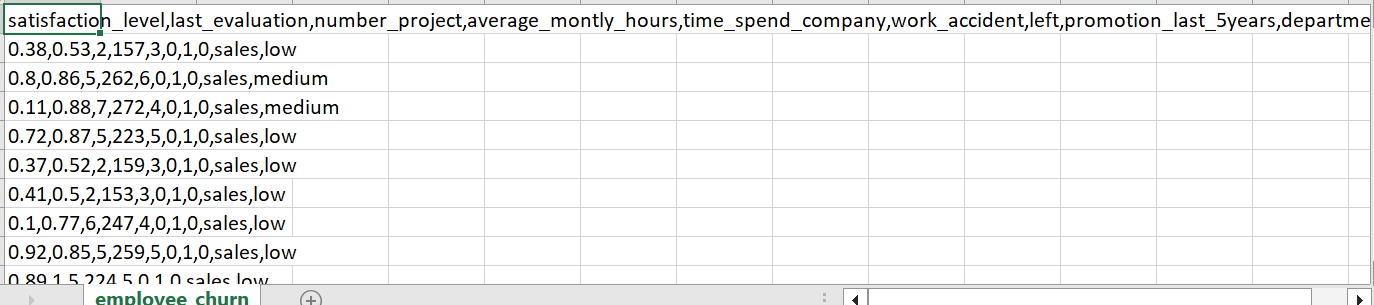
**Вариант 1**



Создаем папку, где будет храниться код и сам размеченный набор данных

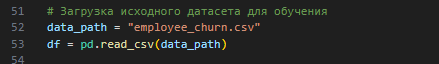


Как выглядит набор данных



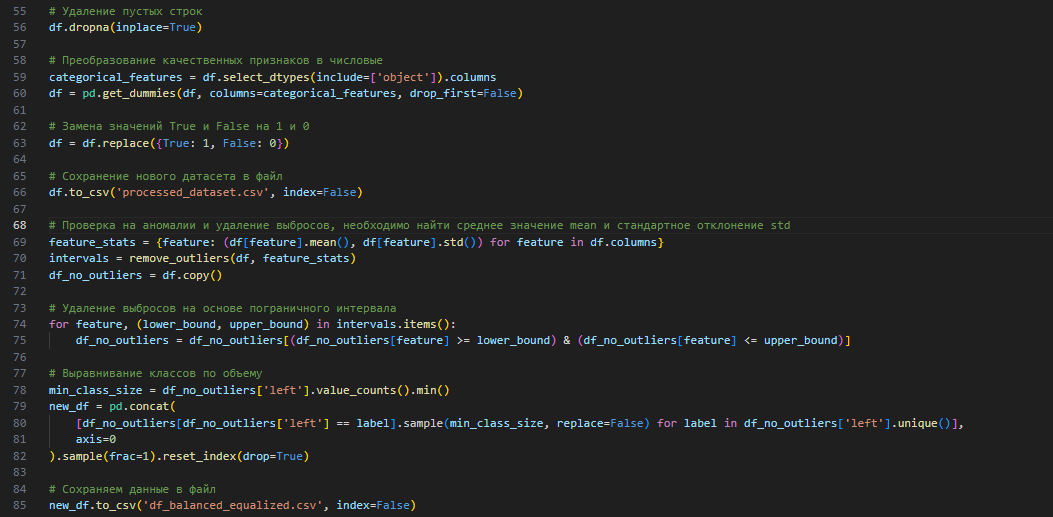
**Написание кода**

Загрузка набора данных и запись в переменную df



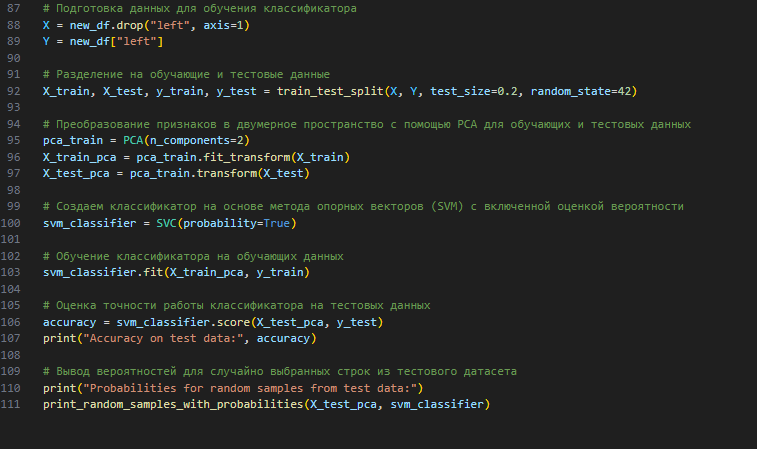
Обработка данных, нам необходимо:

Удалить строки с пропусками, преобразовать все качественные признаки к количественным (в наборе данных это поля salary и department), проверить набор данных на аномалии и удалить их, а также выровнять классы по обьему



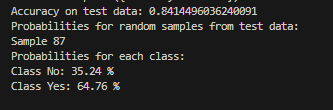
Обучение классификатора

Необходимо разделить данные на обучающие и тестовые, преобразовать данные в двумерное пространство, создать классификатор и оценить его точность, а также вывести принадлежность к тому или иному классу.

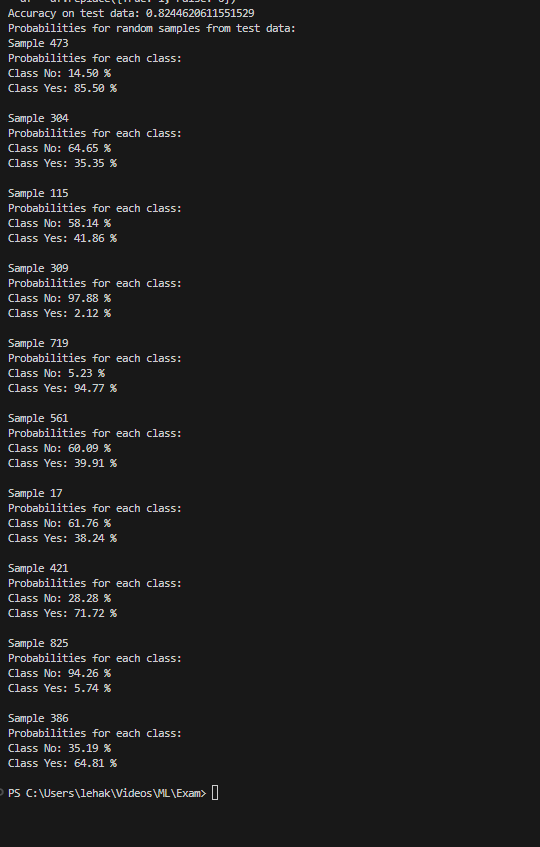


Результат работы кода:

Точность классификатора:



Принадлежность к классам



**Код**

import pandas as pd

from sklearn.svm import SVC

from sklearn.model\_selection import train\_test\_split

from sklearn.decomposition import PCA

import numpy as np

import matplotlib.pyplot as plt

from mlxtend.plotting import plot\_decision\_regions

# Функция для печати случайных примеров с вероятностями

def print\_random\_samples\_with\_probabilities(X, classifier, num\_samples=10):

    # Check if classifier supports probability estimation

    if hasattr(classifier, 'predict\_proba'):

        # Получаем вероятности для всех данных

        probabilities\_all\_data = classifier.predict\_proba(X)

        # Случайно выбираем индексы строк из датасета

        random\_indices = np.random.choice(X.shape[0], num\_samples, replace=False)

        # Выводим информацию о вероятностях для выбранных строк

        for idx in random\_indices:

            print("Sample", idx + 1)

            print("Probabilities for each class:")

            print("Class No:", f"{probabilities\_all\_data[idx][0]\*100:.2f} %")

            print("Class Yes:", f"{probabilities\_all\_data[idx][1]\*100:.2f} %")

            print()

    else:

        print("Probability estimation is not supported by the classifier.")

# Функция для удаления выбросов на основе пограничного интервала

def remove\_outliers(data, feature\_ranges):

    intervals = {}

    for feature, (mean, std) in feature\_ranges.items():

        lower\_bound = mean - 3 \* std

        upper\_bound = mean + 3 \* std

        intervals[feature] = (lower\_bound, upper\_bound)

    # Удаляем столбец 'left' из интервалов

    if 'left' in intervals:

        del intervals['left']

    return intervals

# Функция для детектирования аномалий на основе пограничного интервала

def detect\_anomalies(data, intervals):

    anomalies = []

    for feature, (lower\_bound, upper\_bound) in intervals.items():

        outliers = data[(data[feature] < lower\_bound) | (data[feature] > upper\_bound)]

        anomalies.extend(outliers.index)

    return list(set(anomalies))

# Загрузка исходного датасета для обучения

data\_path = "employee\_churn.csv"

df = pd.read\_csv(data\_path)

# Удаление пустых строк

df.dropna(inplace=True)

# Преобразование качественных признаков в числовые

categorical\_features = df.select\_dtypes(include=['object']).columns

df = pd.get\_dummies(df, columns=categorical\_features, drop\_first=False)

# Замена значений True и False на 1 и 0

df = df.replace({True: 1, False: 0})

# Сохранение нового датасета в файл

df.to\_csv('processed\_dataset.csv', index=False)

# Проверка на аномалии и удаление выбросов, необходимо найти среднее значение mean и стандартное отклонение std

feature\_stats = {feature: (df[feature].mean(), df[feature].std()) for feature in df.columns}

intervals = remove\_outliers(df, feature\_stats)

df\_no\_outliers = df.copy()

# Удаление выбросов на основе пограничного интервала

for feature, (lower\_bound, upper\_bound) in intervals.items():

    df\_no\_outliers = df\_no\_outliers[(df\_no\_outliers[feature] >= lower\_bound) & (df\_no\_outliers[feature] <= upper\_bound)]

# Выравнивание классов по объему

min\_class\_size = df\_no\_outliers['left'].value\_counts().min()

new\_df = pd.concat(

    [df\_no\_outliers[df\_no\_outliers['left'] == label].sample(min\_class\_size, replace=False) for label in df\_no\_outliers['left'].unique()],

    axis=0

).sample(frac=1).reset\_index(drop=True)

# Сохраняем данные в файл

new\_df.to\_csv('df\_balanced\_equalized.csv', index=False)

# Подготовка данных для обучения классификатора

X = new\_df.drop("left", axis=1)

Y = new\_df["left"]

# Разделение на обучающие и тестовые данные

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=42)

# Преобразование признаков в двумерное пространство с помощью PCA для обучающих и тестовых данных

pca\_train = PCA(n\_components=2)

X\_train\_pca = pca\_train.fit\_transform(X\_train)

X\_test\_pca = pca\_train.transform(X\_test)

# Создаем классификатор на основе метода опорных векторов (SVM) с включенной оценкой вероятности

svm\_classifier = SVC(probability=True)

# Обучение классификатора на обучающих данных

svm\_classifier.fit(X\_train\_pca, y\_train)

# Оценка точности работы классификатора на тестовых данных

accuracy = svm\_classifier.score(X\_test\_pca, y\_test)

print("Accuracy on test data:", accuracy)

# Вывод вероятностей для случайно выбранных строк из тестового датасета

print("Probabilities for random samples from test data:")

print\_random\_samples\_with\_probabilities(X\_test\_pca, svm\_classifier)