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.

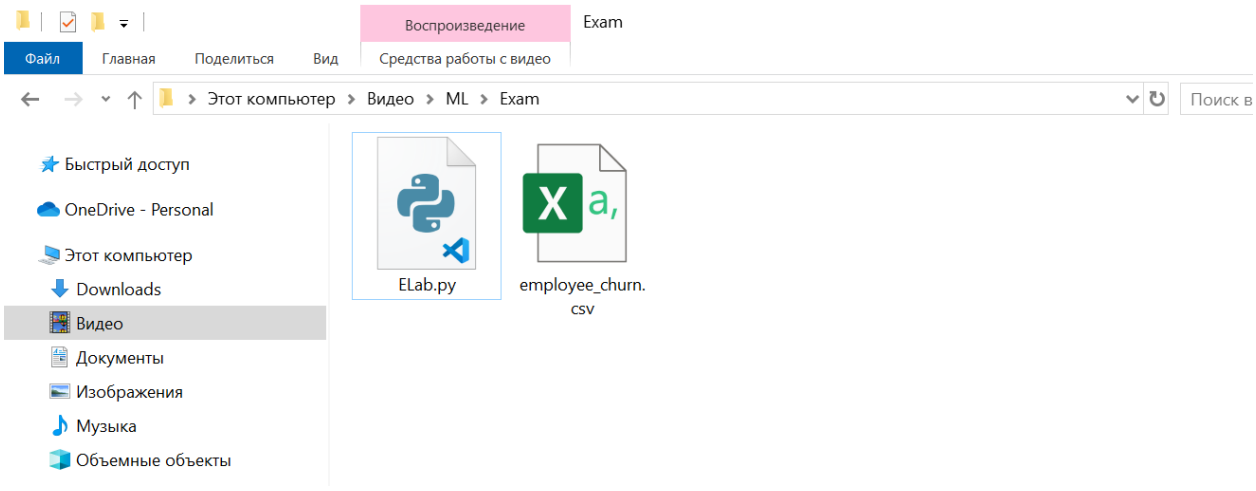
Москва 2024 г.

Вариант 1

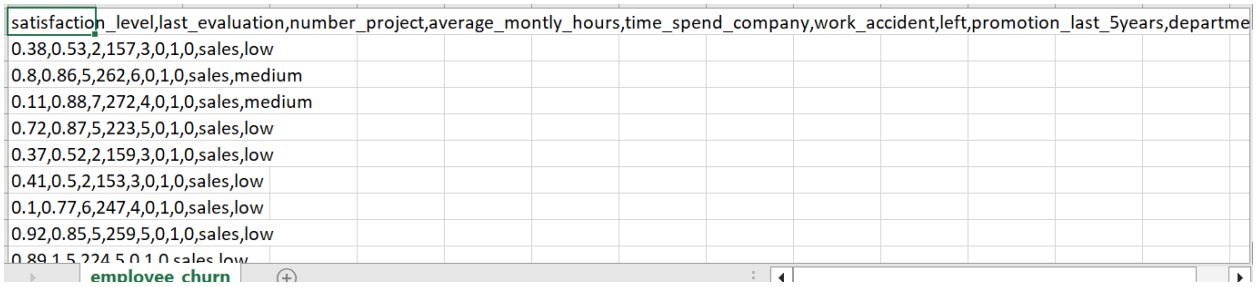
ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 1

1. Имеется размеченный набор данных (employee_churn.csv) с описанием трудовых качеств сотрудников некоторой компании. Признаки объектов: **satisfaction_level** – степень удовлетворенности сотрудника, **last_evaluation** – показатель эффективности сотрудника, **number_project** – число выполненных проектов, **average_montly_hours** – среднемесячное число отработанных часов, **time_spend_company** – стаж работы в компании, **work_accident** – получение производственной травмы, **promotion_last_5years** – повышение в должности за последние пять лет, **department** – отдел, **salary** – уровень зарплаты. Целевой признак – **left**, его значения: **0** – сотрудник работает в настоящее время, **1** – сотрудник уволен. С помощью метода опорных векторов постройте классификатор для прогнозирования категории сотрудника. Оцените качество классификатора. Приведите примеры использования классификатора.

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



Как выглядит набор данных



satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	work_accident	left	promotion_last_5years	departme
0.38	0.53	2	157	3	0	1	0	sales low
0.8	0.86	5	262	6	0	1	0	sales medium
0.11	0.88	7	272	4	0	1	0	sales medium
0.72	0.87	5	223	5	0	1	0	sales low
0.37	0.52	2	159	3	0	1	0	sales low
0.41	0.5	2	153	3	0	1	0	sales low
0.1	0.77	6	247	4	0	1	0	sales low
0.92	0.85	5	259	5	0	1	0	sales low
0.89	1	5	224	5	0	1	0	sales low

employee_churn

Написание кода

Загрузка набора данных и запись в переменную df

```
51  # Загрузка исходного датасета для обучения
52  data_path = "employee_churn.csv"
53  df = pd.read_csv(data_path)
54
```

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

```python
# 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)
```

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.

```python
# 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)
```

Результат работы кода:

Точность классификатора:

```
Accuracy on test data: 0.8414496036240091
Probabilities for random samples from test data:
Sample 87
Probabilities for each class:
Class No: 35.24 %
Class Yes: 64.76 %
```

Принадлежность к классам

```
Accuracy on test data: 0.8244620611551529
Probabilities for random samples from test data:
Sample 473
Probabilities for each class:
Class No: 14.50 %
Class Yes: 85.50 %

Sample 304
Probabilities for each class:
Class No: 64.65 %
Class Yes: 35.35 %

Sample 115
Probabilities for each class:
Class No: 58.14 %
Class Yes: 41.86 %

Sample 309
Probabilities for each class:
Class No: 97.88 %
Class Yes: 2.12 %

Sample 719
Probabilities for each class:
Class No: 5.23 %
Class Yes: 94.77 %

Sample 561
Probabilities for each class:
Class No: 60.09 %
Class Yes: 39.91 %

Sample 17
Probabilities for each class:
Class No: 61.76 %
Class Yes: 38.24 %

Sample 421
Probabilities for each class:
Class No: 28.28 %
Class Yes: 71.72 %

Sample 825
Probabilities for each class:
Class No: 94.26 %
Class Yes: 5.74 %

Sample 386
Probabilities for each class:
Class No: 35.19 %
Class Yes: 64.81 %

PS C:\Users\lehak\Videos\ML\Exam>
```

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 'attack' 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)
```