

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

«Методы искусственного интеллекта»

Отчёт по лабораторной работе №5

Вариант №7

Выполнил:

студент группы ИСТбд-42

Егоров Иван

Проверил:

доцент кафедры ИВК, к.т.н.

Шишкин В.В.

Ульяновск
2022

Задание 1.

“Ознакомится с классификаторами библиотеки Skikit-learn”.

Ознакомиться с классификаторами библиотеки Scikit-learn.

Результат.

Мы ознакомились с классификаторами библиотеки Scikit-learn.

Задание 2.

“Выбрать для исследования не менее трёх классификаторов”.

Необходимо выбрать для исследования не менее трёх классификаторов.

Результат.

Нами были выбраны такие классификаторы:

- Перцептрон
- Дерева решений
- Метод опорных векторов

Задание 3.

“Выбрать набор данных для задач классификации из открытых источников”.

Из перечисленных источников необходимо выбрать набор данных для задач классификации:

- <https://tproger.ru/translations/the-best-datasets-for-machine-learning-and-data-science/>
- <https://vc.ru/ml/150241-15-proektov-dlya-razvitiya-navykov-raboty-s-mashinnym-obucheniem>
- <https://archive.ics.uci.edu/ml/index.php>
- <https://habr.com/ru/company/edison/blog/480408/>
- <https://www.kaggle.com/datasets/>

Результат.

Нами был выбран dataset с сайта <https://www.kaggle.com/datasets/> :
InfoSec/Cyber Security Salaries Classification.

<https://www.kaggle.com/datasets/whenamancodes/infosec-cyber-security-salaries>

The screenshot shows the Kaggle interface for a dataset titled "InfoSec/Cyber Security Salaries Classification" by AMAN CHAUHAN. The dataset is described as "Salaries of Different InfoSec/Cyber Security Fields". The page includes a sidebar with navigation options like Home, Competitions, Datasets, Code, Discussions, Learn, and More. The main content area has tabs for Data Card, Code (2), and Discussion (0). The "About Dataset" section contains a table with dimensions and their descriptions.

Dimension	Description
Working Year	The year the salary was paid
Job Title	The role worked in during the year
Experience Level	The experience level in the job during the year. [EN - Entry level / Junior, MI - Mid level / Intermediate, SE - Senior level / Expert, EX - Executive level / Director]
Job status	The type of employment for the role. [PT - Part time, FT - Full time, CT - Contract, FL - Freelance]
Salary	The total gross salary amount paid.

Additional metadata on the right includes Usability (10.00), License (CC0: Public Domain), and Expected update frequency (Never).

Задание 4.

“Выбор классификаторов и набора данных утвердить у преподавателя”.

Выбор классификаторов и набора данных утвердить у преподавателя.

Результат.

Выбор классификаторов и набора данных мы утвердили у преподавателя.

Задание 5.

“Для каждого классификатора определить целевой столбец и набор признаков. Обосновать свой выбор. При необходимости преобразовать типы признаков данных”.

Для каждого классификатора определить целевой столбец и набор признаков. Обосновать свой выбор. При необходимости преобразовать типы признаков данных.

Результат.

Для всех классификаторов мы выбрали столбец `experience_level` в качестве целевого т.к. от оставшихся признаков будет зависеть уровень опыта сотрудника.

Признаковые данные мы преобразовали в числовые.

```

#Подготавливаем данные
def preparation(df):
    df = df.copy()
    df = df.drop('salary',axis=1)
    df = df.drop('salary_currency', axis=1)
    df['employment_type'] = pandas.factorize(df['employment_type'])[0]
    df['job_title'] = pandas.factorize(df['job_title'])[0]
    df['employee_residence'] = pandas.factorize(df['employee_residence'])[0]
    df['company_location'] = pandas.factorize(df['company_location'])[0]
    df['company_size'] = pandas.factorize(df['company_size'])[0]
    X = df.drop('experience_level', axis=1)
    y = df['experience_level']
    X = pandas.DataFrame(X, index=X.index, columns=X.columns)
    return X, y

```

Задание 6.

“Подготовить данные к обучению”.

Подготовить данные к обучению.

Результат.

Мы подготовили данные для обучения.

```

#Подготавливаем данные
def preparation(df):
    df = df.copy()
    df = df.drop('salary',axis=1)
    df = df.drop('salary_currency', axis=1)
    df['employment_type'] = pandas.factorize(df['employment_type'])[0]
    df['job_title'] = pandas.factorize(df['job_title'])[0]
    df['employee_residence'] = pandas.factorize(df['employee_residence'])[0]
    df['company_location'] = pandas.factorize(df['company_location'])[0]
    df['company_size'] = pandas.factorize(df['company_size'])[0]
    X = df.drop('experience_level', axis=1)
    y = df['experience_level']
    X = pandas.DataFrame(X, index=X.index, columns=X.columns)
    return X, y

```

```

#Масштабируем данные
scaler = StandardScaler()
X_teach, y_teach = preparation(teach_df)
X_teach = scaler.fit_transform(X_teach)
X_test, y_test = preparation(test_df)
X_test = scaler.fit_transform(X_test)

```

Задание 7-8.

“Провести обучение и оценку моделей на сырых данных”. “Провести предобработку данных”.

Провести обучение и оценку моделей на сырых данных. Провести предобработку данных.

Результат.

Данные пункты мы пропускаем т.к. данные уже очищены.

Задание 9.

“Провести обучение и оценку моделей на очищенных данных”.

Провести обучение и оценку моделей на очищенных данных.

Результат.

Мы провели обучение и оценку моделей на очищенных данных.

```
#Прогоняем тестовые данные через классификаторы и выводим точность предсказаний

#Перцептрон
mlp = MLPClassifier(random_state = 1, max_iter = 300).fit(X_train, y_train)
mlp_predictions = pandas.Series(mlp.predict(X_test))
print('Точность предсказаний многослойного перцептрона: ' + str(mlp.score(X_test, y_test)*100) + '%')

#Дерева решений
dtc = DecisionTreeClassifier(random_state = 0).fit(X_train, y_train)
dtc_predictions = pandas.Series(dtc.predict(X_test))
print('Точность предсказаний дерева решений: ' + str(dtc.score(X_test, y_test)*100) + '%')

#Метод опорных векторов
svm = SVC(kernel = 'rbf').fit(X_train, y_train)
svm_predictions = pandas.Series(svm.predict(X_test))
print('Точность предсказаний методом опорных векторов: ' + str(svm.score(X_test, y_test)*100) + '%')
```

Точность предсказаний многослойного перцептрона: 67.97627872498147%

Точность предсказаний дерева решений: 98.88806523350631%

Точность предсказаний методом опорных векторов: 66.04892512972572%

За

Задание 10.

“Проанализировать результаты”.

Проанализировать результаты.

Результат.

С имеющимися данными высокую эффективность показал классификатор дерева решений. Многослойный перцептрон и метод опорных векторов показали эффективность низкую в два раза по сравнению с деревом решений.

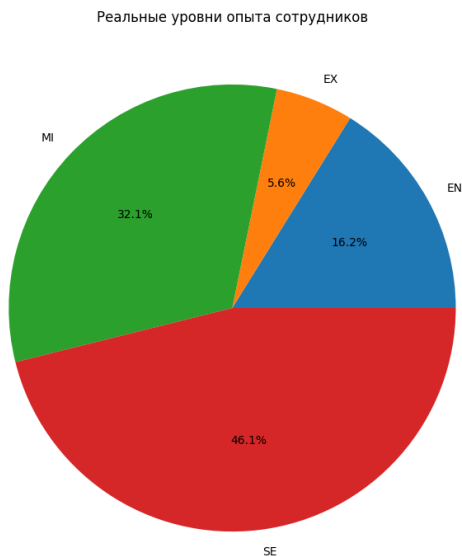
Задание 11.

“Результаты анализа предоставить в табличной и графической форме”.

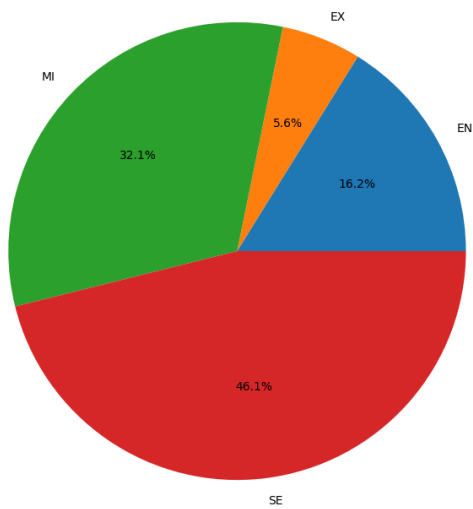
Результаты анализа предоставить в табличной и графической форме.

Результат.

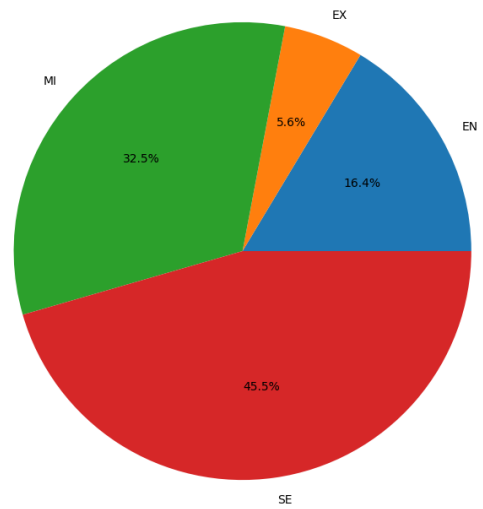
Мы предоставили результаты анализа в графической форме.



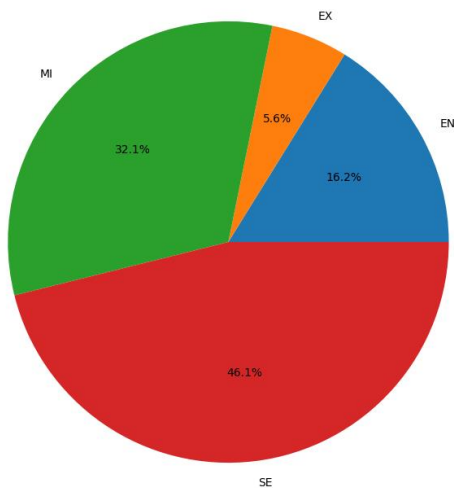
Реальные уровни опыта сотрудников



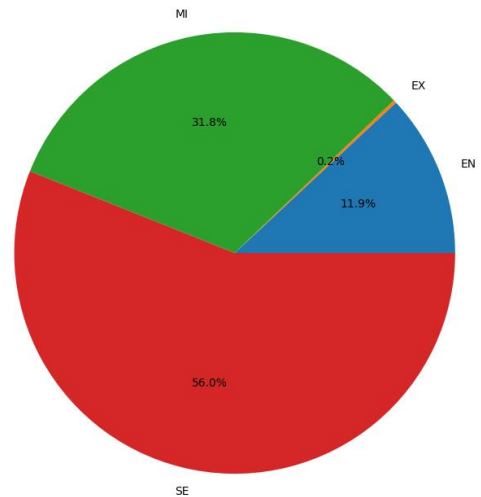
Уровни опыта сотрудников предсказанные деревом решений



Реальные уровни опыта сотрудников



Уровни опыта сотрудников предсказанные методом опорных векторов



Задание 12.

“Сформулировать выводы”.

Сформулировать выводы.

Результат.

В ходе проведённой нами работы мы провели классификацию выбранного набора данных тремя методами классификации. В результате мы выявили , что среди : многослойного перцептрона , метода опорных векторов и дерева решений; дерево решений является самым эффективным методом классификации с учётом имеющихся данных.

Код.

```
import pandas
import matplotlib.pyplot as pyplot
import pylab
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier

#Читаем данные из csv
teach_df = pandas.read_csv('Cyber_salaries.csv')
test_df = teach_df

#Подготавливаем данные
def preparation(df):
    df = df.copy()
    df = df.drop('salary',axis=1)
    df = df.drop('salary_currency', axis=1)
    df['employment_type'] = pandas.factorize(df['employment_type'])[0]
    df['job_title'] = pandas.factorize(df['job_title'])[0]
    df['employee_residence'] = pandas.factorize(df['employee_residence'])[0]
    df['company_location'] = pandas.factorize(df['company_location'])[0]
    df['company_size'] = pandas.factorize(df['company_size'])[0]
    X = df.drop('experience_level', axis=1)
    y = df['experience_level']
    X = pandas.DataFrame(X, index=X.index, columns=X.columns)
    return X, y

#Масштабируем данные
scaler = StandardScaler()
X_teach, y_teach = preparation(teach_df)
X_teach = scaler.fit_transform(X_teach)
X_test, y_test = preparation(test_df)
X_test = scaler.fit_transform(X_test)

#Прогоняем тестовые данные через классификаторы и выводим точность предсказаний

#Перцептрон
mlp = MLPClassifier(random_state = 1, max_iter = 300).fit(X_teach, y_teach)
mlp_predictions = pandas.Series(mlp.predict(X_test))
print('Точность предсказаний многослойного перцептрона: ' + str(mlp.score(X_test, y_test)*100) + '%')

#Дерева решений
dtc = DecisionTreeClassifier(random_state = 0).fit(X_teach, y_teach)
dtc_predictions = pandas.Series(dtc.predict(X_test))
print('Точность предсказаний дерева решений: ' + str(dtc.score(X_test, y_test)*100) + '%')

#Метод опорных векторов
svm = SVC(kernel = 'rbf').fit(X_teach, y_teach)
svm_predictions = pandas.Series(svm.predict(X_test))
print('Точность предсказаний методом опорных векторов: ' + str(svm.score(X_test, y_test)*100) + '%')

#Отрисовываем графики с предсказанными и реальными значениями уровней опыта сотрудников

#Перцептрон
pylab.figure(figsize=(20,10))
pylab.subplot(1, 2, 1)
pyplot.pie(y_test.value_counts().sort_index(), labels = sorted(y_test.unique()), autopct='%1.1f%%')
```



```

pyplot.title('Реальные уровни опыта сотрудников')
pylab.subplot(1, 2, 2)
pyplot.pie(mlp_predictions.value_counts().sort_index(), labels = sorted(mlp_predictions.unique()),
autopct='%1.1f%%')
pyplot.title('Уровни опыта сотрудников предсказанные многослойным перцептроном')
pyplot.show()

```

```

#Дерева решений
pylab.figure(figsize=(20,10))
pylab.subplot(1, 2, 1)
pyplot.pie(y_test.value_counts().sort_index(), labels = sorted(y_test.unique()), autopct='%1.1f%%')
pyplot.title('Реальные уровни опыта сотрудников')
pylab.subplot(1, 2, 2)
pyplot.pie(dtc_predictions.value_counts().sort_index(), labels = sorted(dtc_predictions.unique()),
autopct='%1.1f%%')
pyplot.title('Уровни опыта сотрудников предсказанные деревом решений')
pyplot.show()

```

```

#Метод опорных векторов
pylab.figure(figsize=(20,10))
pylab.subplot(1, 2, 1)
pyplot.pie(y_test.value_counts().sort_index(), labels = sorted(y_test.unique()), autopct='%1.1f%%')
pyplot.title('Реальные уровни опыта сотрудников')
pylab.subplot(1, 2, 2)
pyplot.pie(svm_predictions.value_counts().sort_index(), labels = sorted(svm_predictions.unique()),
autopct='%1.1f%%')
pyplot.title('Уровни опыта сотрудников предсказанные методом опорных векторов')
pyplot.show()

```