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.

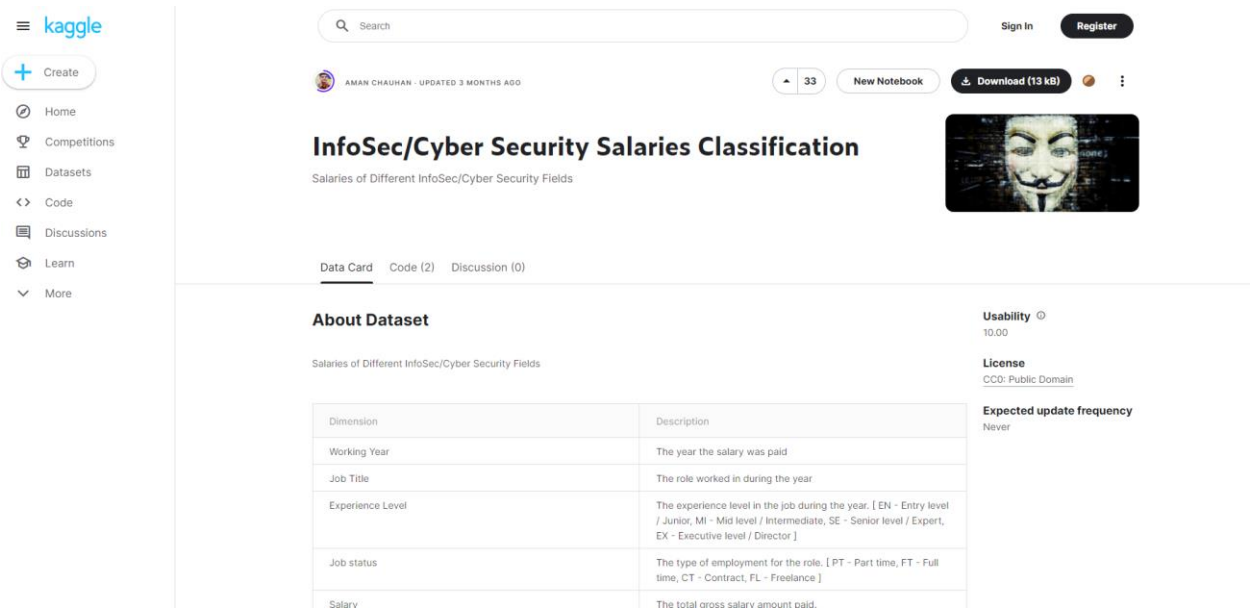
Нами были выбраны такие классификаторы:
-Перцептрон
-Дерева решений
-Метод опорных веторов

**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/infoseccyber-security-salaries




**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 признаковых данных”.

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

```python
#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.

```python
#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
```

```python
#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_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) + '%')
```

```
Точность предсказаний многослойного перцептрона: 43.47181008902077%
Точность предсказаний дерева решений: 33.82789317507419%
Точность предсказаний методом опорных векторов: 42.433234421364986%
```

**Задание 10.**
“Проанализировать результаты”.

Проанализировать результаты.

Результат.

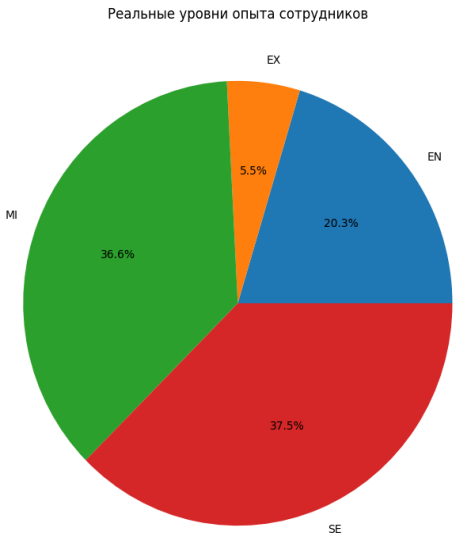
С имеющимися данными высокую эффективность показал классификатор дерева решений. Многослойный перцептрон и метод опорных векторов показали эффективность низкую в два раза по сравнению с деревом решений.

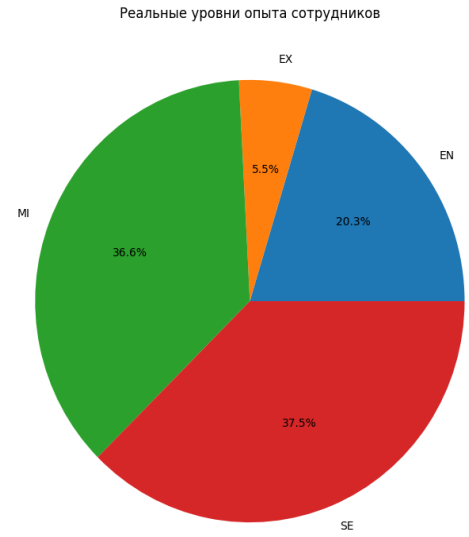
**Задание 11.**
“Результаты анализа предоставить в табличной и графической форме”.

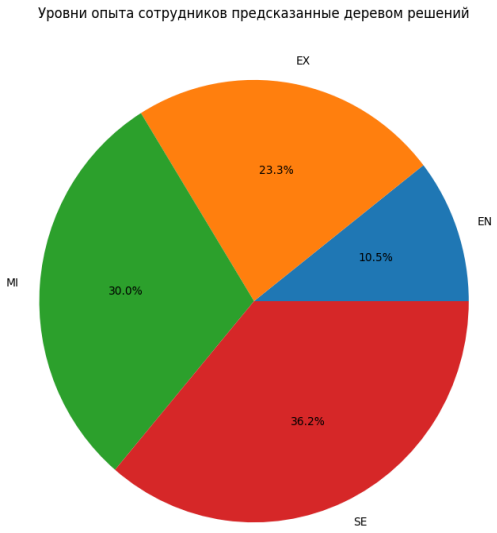
Результаты анализа предоставить в табличной и графической форме.

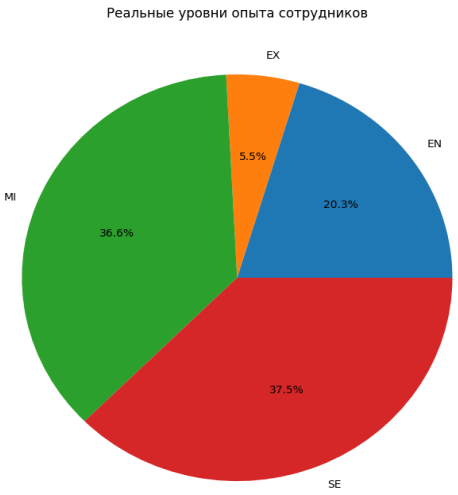
Результат.

Мы предоставили результаты анализа в графической форме.





**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; многослойный перцептрон 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
df = pandas.read_csv('Cyber_salaries.csv')
teach_df = df.iloc [:675]
test_df = df.iloc [675:]

#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()
```