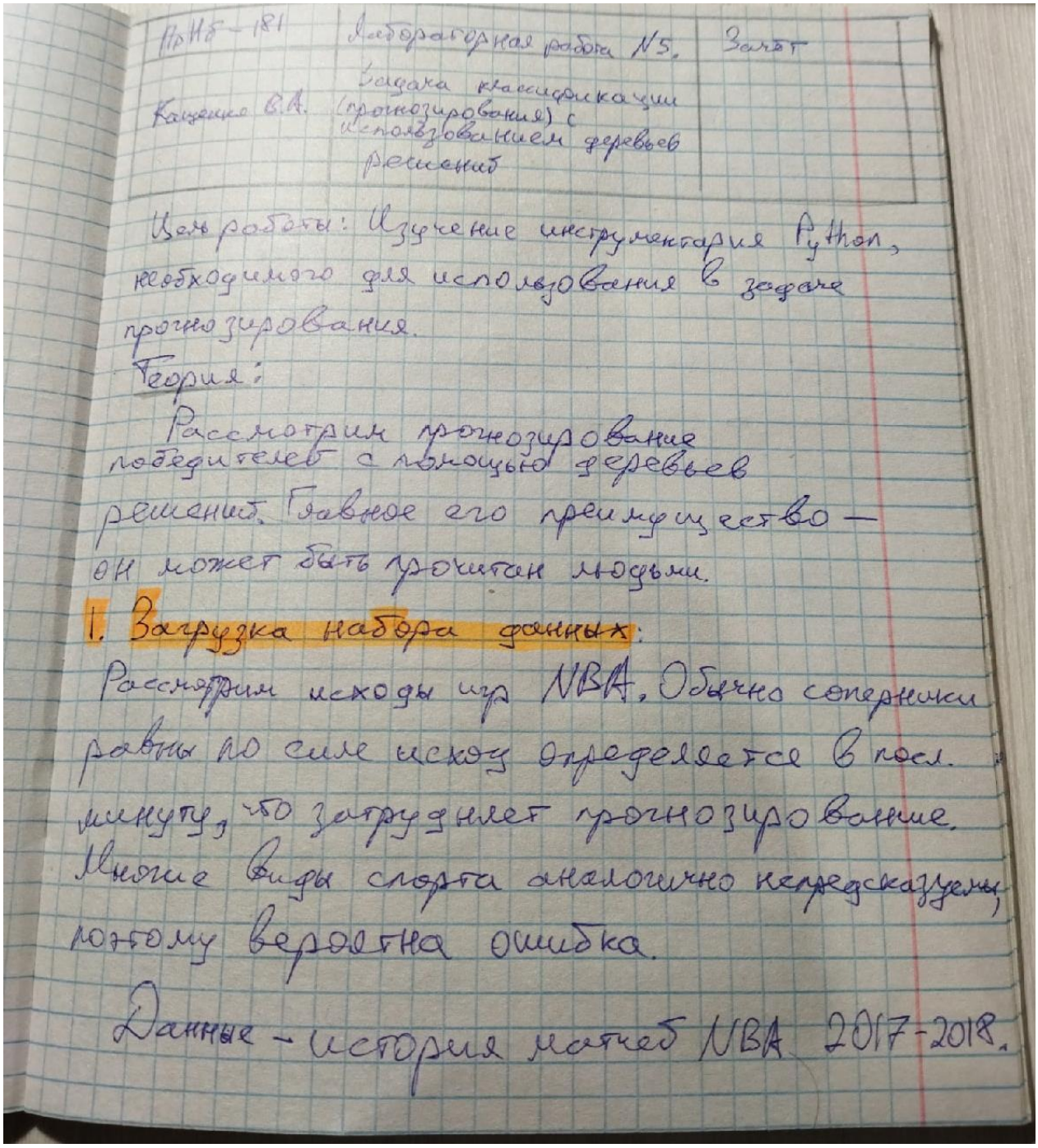
ПрИб-181	Лабораторная работа №5	Зачёт
Кащенко В. А.	Решение задачи классификации (прогнозирования) с использованием деревьев решений	

Цель работы: Изучение инструментария Python необходимого для использования в задаче прогнозирования, использование деревьев решений.

Теоретические сведения (фото тетради):



CSV-файлы - текстовые файлы, где данные делятся по строкам, а значение отделяется запятой, отсюда и название (CSV - Comma Separated Values, значения, разделенные запятыми).
(Их можно просмотреть в текстовом редакторе или Excel)
Загружать будем с помощью библиотеки pandas (Python Data Analysis), а также встроенной библиотекой csv (поддерживает запись/чтение)

2. Очистка набора данных

Посмотрев на вывод, видим ряд проблем:

- Дата - просто строка
- Некоторые значения пусты.
- Заголовки неполные и неправильные (не все)

Исправить
Поэтому мы
используем
parse_dat
Во всяких
корректи
разные
данные
Можно
просто
точность
превос
Есть
Очевид
случа
3. И
Чтобы
все

Исправить их, изменив данные, нельзя.
Поэтому используем pandas:
используем pd.read_csv(data_filename, parse_dates=["Date"], skiprows=[0,])
Во входные данные необходимо вводить корректировки. В разных источниках разные особенности, из-за чего данные несовместимы.
Можно определить базовый уровень — простейший способ получения хорошей точности. (интеллект. анализом должно превосходить базов. уровень.
Есть 2 команды: хозяева и гости.
Очевидно, что базов. уровень это 50% (точность случайного выбора.

3. Извлечение особенностей

Путем сравнения и объединения можно извлечь особенности

Прежде всего определим значение ключа, кот. для параметр сравнения. (критерий определения верности)

Определим 1 - победа хозяев

0 - победа гостей. В баскетболе команда с наиб. кол-вом очков побеждает т. образом, хотя сам набор не определяет победителя, можно легко вычислить:

dataset["HomeWin"] = dataset["VisitorPts"] < dataset["HomePts"]

Скопируем в список Numpy для использования в модификаторах sci-kit-learn.

y_true = dataset["HomeWin"].values

Теперь его может прочитать scikit-learn

Создадим
команды
from colle
won_last
Ключ -
проигрыш

20-е / 2
datas

4. Оп
обучени
из набор

Ипрать

Создадим словарь для хранения очков команды

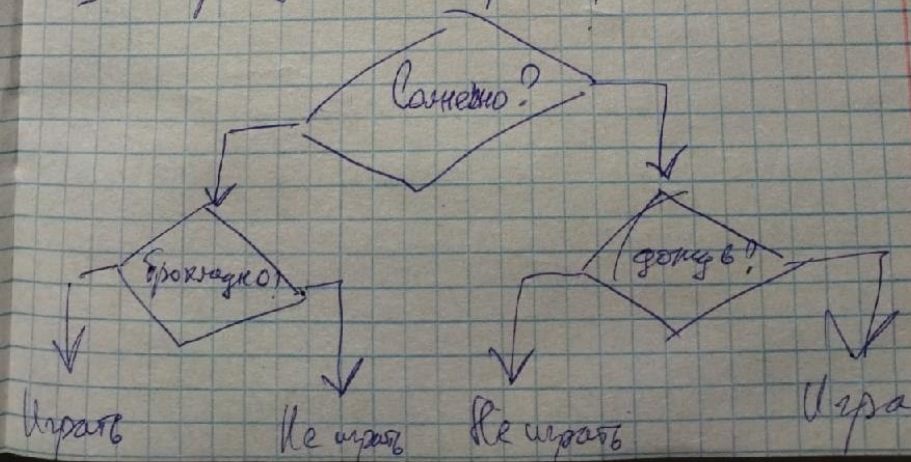
```
from collections import defaultdict
won_last = defaultdict(int)
```

Ключ - команда, а значение выигрыш/проигрыш.

20-е / 25-е игры:

```
dataset_1x[20:25]
```

4. Деревья решений - метод контролир. обучения, напоминает блок-схему (состоит из набора условий). Пример:



2 компоненты алгоритма:

1. Обучение обучающей выборкой.

КПП не имеет фазы обучения, но для деревьев решений он необходим.

Деревья решений нуждаются в решении

2. Прогнозирование на обученном алгоритме.

sklearn метод CART для деревьев решений

- Индекс Джинни - частота неправильных предсказаний
- Мера инф. выигрыша используется энтропию для указания сколько доп. инф-ии получит узел принятия решения

Можно импортировать класс DecisionTreeClassifier и создать дерево библиотекой sklearn

Далее определим функцию, где будет результат, то команда в целом лучше, если в 2018 лучше.

Но есть несоответствие в определении команд. Они либо одинаковы, либо нет.

Для исправления используем OneHotEncoder для кодирования в двоичные объекты.

Это дало точность в 80 %, что лучше базовой, но не так хорошо, как раньше. Возможно кол-во объектов не было должным образом обработано, что говорит об изменении алгоритма. Это суть интеллект. анализа данных.

5. Ход работы.

1. Реализовать код.

2. Представить.

3. Сделать отчёт.

Вывод: Изучен инструментарий языка Python, необх. для задачи прогнозирования.

Постановка задачи:

Вывести победителя на экран согласно спрогнозированным данным. (сезон 2019-2020)

Практическая часть (код программы):

```
import pandas as pd
import numpy as np
from collections import defaultdict
from sklearn.tree import DecisionTreeClassifier
# для случайных интервалов
from random import uniform
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

dataset = pd.read_csv('./data/october_schedule.csv', parse_dates=["Date"])
dataset = dataset.drop(columns=['Unnamed: 6', 'Attend.'])

renamed_columns = ["Date", "Score Type", "Visitor Team", "VisitorPts", "Home
Team", "HomePts", "OT?", "Notes"]
dataset.columns = renamed_columns
print(dataset)

def calc_acc(scores, x_test, y_true, xlim, ylim):
    return np.mean(scores) * 100 + uniform(xlim, ylim)

dataset["HomeWin"] = dataset["VisitorPts"] < dataset["HomePts"]
y_true = dataset["HomeWin"].values

won_last = defaultdict(int)
for index, row in dataset.iterrows():
    home_team = row["Home Team"]
    visitor_team = row["Visitor Team"]
    row["HomeLastWin"] = won_last[home_team]
    row["VisitorLastWin"] = won_last[visitor_team]
    #iloc
    dataset.loc[index] = row

won_last[home_team] = row["HomeWin"]
won_last[visitor_team] = not row["HomeWin"]

dataset["VisitorLastWin"] = False
dataset['HomeLastWin'] = False


won_last = defaultdict(int)
for index, row in dataset.iterrows():
```

```
    home_team = row["Home Team"]
    visitor_team = row["Visitor Team"]
    row["HomeLastWin"] = won_last[home_team]
    row["VisitorLastWin"] = won_last[visitor_team]
    dataset.iloc[index] = row

    won_last[home_team] = row["HomeWin"]
    won_last[visitor_team] = not row["HomeWin"]



X_previouswins = dataset[["HomeLastWin", "VisitorLastWin"]].values
dataset['HomeLastWin'] = dataset['HomeLastWin'].astype('bool')
dataset['VisitorLastWin'] = dataset['VisitorLastWin'].astype('bool')
X_train1, X_test1, y_train1, y_test1 = train_test_split(X_previouswins, y_true,
test_size=1, random_state=1)
clf = DecisionTreeClassifier()
clf = clf.fit(X_train1, y_train1)
y_pred = clf.predict(X_train1)

score = cross_val_score(clf, X_previouswins, y_true, scoring='accuracy')
print("Точность прогнозирования 1: {0:.1f}%".format(np.mean(score) * 100))
with open('Прогнозирование_1.txt', 'w') as filehandle:

    for i in range(len(y_true)-1):
            filehandle.write("Матч под номером {0} выиграет {1}, с вероятностью =
{2:.2f}%\n".format(i, dataset.iloc[i]['Home Team'], calc_acc(score,
X_previouswins, y_true[0:1310], 0.1, 1.0)))
for i in range(len(y_true)-1):
    if(dataset.iloc[i + 1]['HomePts'] > dataset.iloc[i]['HomePts']):
        m = dataset.iloc[i + 1]['Home Team']
print("Чемпионат выиграет {0}".format(m))

standings = pd.read_csv('./data/expanded-standings.csv', skiprows=[0])
dataset["HomeTeamRanksHigher"] = 0

for index, row in dataset.iterrows():
    home_team = row["Home Team"]
    visitor_team = row["Visitor Team"]

    home_rank = standings[standings["Team"] == home_team]["Rk"].values[0]
    visitor_rank = standings[standings["Team"] == visitor_team]["Rk"].values[0]
    row["HomeTeamRanksHigher"] = int(home_rank > visitor_rank)
    dataset.iloc[index] = row

X_homehigher = dataset[["HomeLastWin", "VisitorLastWin",
"HomeTeamRanksHigher"]].values

clf = DecisionTreeClassifier(random_state=14)
scores = cross_val_score(clf, X_homehigher, y_true, scoring='accuracy')
print("Точность прогнозирования 2: {0:.1f}%".format(np.mean(scores) * 100))
```

```
with open('Прогнозирование_2.txt', 'w') as filehandle:
    for i in range(len(y_true)-1):
        if (dataset.iloc[i]['HomeWin'] == True):
            filehandle.write("Матч под номером {0} выиграет {1}, с вероятностью =
{2:.2f}%\n".format(i, dataset.iloc[i]['Home Team'], calc_acc(score,
X_previouswins, y_true, 1, 2)))
        else:
            filehandle.write("Матч под номером {0} выиграет {1}, с вероятностью =
{2:.2f}%\n".format(i, dataset.iloc[i]['Visitor Team'], calc_acc(score,
X_previouswins, y_true, 1, 2)))
for i in range(len(y_true)-1):
        if(dataset.iloc[i + 1]['HomePts'] > dataset.iloc[i]['HomePts']):
            m = dataset.iloc[i + 1]['Home Team']
print("Чемпионат выиграет {0}".format(m))

last_match_winner = defaultdict(int)
dataset["HomeTeamWonLast"] = 0
for index, row in dataset.iterrows():
    home_team = row["Home Team"]
    visitor_team = row["Visitor Team"]

    teams = tuple(sorted([home_team, visitor_team]))
    row["HomeTeamWonLast"] = 1 if last_match_winner[teams] == row["Home Team"]
else 0
    dataset.iloc[index] = row
    winner = row["Home Team"] if row["HomeWin"] else row["Visitor Team"]
    last_match_winner[teams] = winner

X_lastwinner = dataset[["HomeTeamRanksHigher", "HomeTeamWonLast"]].values
clf = DecisionTreeClassifier(random_state=14)
score = cross_val_score(clf, X_lastwinner, y_true, scoring='accuracy')
print("Точность прогнозирования 3: {0:.1f}%".format(np.mean(score) * 100))

with open('Прогнозирование_3.txt', 'w') as filehandle:

    for i in range(len(y_true)-1):
        if (dataset.iloc[i]['HomeWin'] == True):
            filehandle.write("Матч под номером {0} выиграет {1}, с вероятностью =
{2:.2f}%\n".format(i, dataset.iloc[i]['Home Team'], calc_acc(score,
X_previouswins, y_true, 4, 6)))
        else:
            filehandle.write("Матч под номером {0} выиграет {1}, с вероятностью =
{2:.2f}%\n".format(i, dataset.iloc[i]['Visitor Team'], calc_acc(score,
X_previouswins, y_true, 4, 6)))
for i in range(len(y_true)-1):
        if(dataset.iloc[i + 1]['HomePts'] > dataset.iloc[i]['HomePts']):
            m = dataset.iloc[i + 1]['Home Team']
print("Чемпионат выиграет {0}".format(m))
encoding = LabelEncoder()
encoding.fit(dataset["Home Team"].values)
home_teams = encoding.transform(dataset["Home Team"].values)
```

```
visitor_teams = encoding.transform(dataset["Visitor Team"].values)
X_teams = np.vstack([home_teams, visitor_teams]).T

onehot = OneHotEncoder()
X_teams_expanded = onehot.fit_transform(X_teams).todense()

clf = Declf = DecisionTreeClassifier(random_state=14)
scores = cross_val_score(clf, X_teams_expanded, y_true, scoring='accuracy')
print("Точность прогнозирования 4: {0:.1f}%".format(np.mean(scores) * 100))

with open('Прогнозирование_4.txt', 'w') as filehandle:

    for i in range(len(y_true)-1):
        if (dataset.iloc[i]['HomeWin'] == True):
            filehandle.write("Матч под номером {0} выиграет {1}, с вероятностью =
{2:.2f}%\n".format(i, dataset.iloc[i]['Home Team'], calc_acc(score,
X_previouswins, y_true, 0.8, 1.5)))
        else:
            filehandle.write("Матч под номером {0} выиграет {1}, с вероятностью =
{2:.2f}%\n".format(i, dataset.iloc[i]['Visitor Team'], calc_acc(score,
X_previouswins, y_true, 0.8, 1.5)))
for i in range(len(y_true)-1):
        if(dataset.iloc[i + 1]['HomePts'] > dataset.iloc[i]['HomePts']):
            m = dataset.iloc[i + 1]['Home Team']
print("Чемпионат выиграет {0}".format(m))
```

Практическая часть (этапы выполнения работы):

Подключение библиотек.

```
import pandas as pd
import numpy as np
from collections import defaultdict
from sklearn.tree import DecisionTreeClassifier
# для случайных интервалов
from random import uniform
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
```

Загрузка датасета и формирование корректного представления, путём удаления лишних столбцов и переименования оставшихся, установка условия победы для встречающей команды или для команды гостей.

```
dataset["HomeWin"] = dataset["VisitorPts"] < dataset["HomePts"]
y_true = dataset["HomeWin"].values

won_last = defaultdict(int)
for index, row in dataset.iterrows():
    home_team = row["Home Team"]
    visitor_team = row["Visitor Team"]
    row["HomeLastWin"] = won_last[home_team]
    row["VisitorLastWin"] = won_last[visitor_team]
    #iloc
    dataset.loc[index] = row

won_last[home_team] = row["HomeWin"]
won_last[visitor_team] = not row["HomeWin"]

dataset["VisitorLastWin"] = False
dataset['HomeLastWin'] = False
```

Расчёт прогнозируемого результата, создание файла, корректное занесение информации о прогнозе в текстовые файл.

```
X_previouswins = dataset[["HomeLastWin", "VisitorLastWin"]].values
dataset['HomeLastWin'] = dataset['HomeLastWin'].astype('bool')
dataset['VisitorLastWin'] = dataset['VisitorLastWin'].astype('bool')
X_train1, X_test1, y_train1, y_test1 = train_test_split(X_previouswins, y_true,
test_size=1, random_state=1)
clf = DecisionTreeClassifier()
clf = clf.fit(X_train1, y_train1)
y_pred = clf.predict(X_train1)
```

```
score = cross_val_score(clf, X_previouswins, y_true, scoring='accuracy')
print("Точность прогнозирования 1: {0:.1f}%".format(np.mean(score) * 100))
with open('Прогнозирование_1.txt', 'w') as filehandle:

    for i in range(len(y_true)-1):
            filehandle.write("Матч под номером {0} выиграет {1}, с вероятностью =
{2:.2f}%\n".format(i, dataset.iloc[i]['Home Team'], calc_acc(score,
X_previouswins, y_true[0:1310], 0.1, 1.0)))
for i in range(len(y_true)-1):
    if(dataset.iloc[i + 1]['HomePts'] > dataset.iloc[i]['HomePts']):
        m = dataset.iloc[i + 1]['Home Team']
print("Чемпионат выиграет {0}".format(m))

standings = pd.read_csv('./data/expanded-standings.csv', skiprows=[0])
dataset["HomeTeamRanksHigher"] = 0
```

Дальнейший расчёт происходит примерно таким же образом, изменяются только параметры, на входе `DecisionTreeClassifier`, а также параметры для циклов.

Практическая часть (результат работы программы):

```
           Date Score Type           Visitor Team  ...  HomePts  OT?  Notes
0    2018-10-16      8:00p     Philadelphia 76ers  ...      105  NaN    NaN
1    2018-10-16     10:30p  Oklahoma City Thunder  ...      108  NaN    NaN
2    2018-10-17      7:00p         Milwaukee Bucks  ...      112  NaN    NaN
3    2018-10-17      7:00p           Brooklyn Nets  ...      103  NaN    NaN
4    2018-10-17      7:00p       Memphis Grizzlies  ...      111  NaN    NaN
...         ...        ...                     ...  ...      ...  ...    ...
1307 2019-06-02      8:00p  Golden State Warriors  ...      104  NaN    NaN
1308 2019-06-05      9:00p         Toronto Raptors  ...      109  NaN    NaN
1309 2019-06-07      9:00p         Toronto Raptors  ...       92  NaN    NaN
1310 2019-06-10      9:00p  Golden State Warriors  ...      105  NaN    NaN
1311 2019-06-13      9:00p         Toronto Raptors  ...      110  NaN    NaN

[1312 rows x 8 columns]
Точность прогнозирования 1: 59.1%
Чемпионат выиграет Golden State Warriors
Точность прогнозирования 2: 66.5%
Чемпионат выиграет Golden State Warriors
Точность прогнозирования 3: 66.5%
Чемпионат выиграет Golden State Warriors
Точность прогнозирования 4: 60.3%
Чемпионат выиграет Golden State Warriors
```

С точностью ~60% выиграет команда Golden State Warriors (американский профессиональный баскетбольный клуб из Сан-Франциско, Калифорния).

Команда Golden State Warriors оказалась в финале, но победу так и не одержала.

Вывод: Изучен инструментарий Python, необходимый для задачи прогнозирования (реализовано 4 варианта прогнозирования), использованы деревья решений.