Московский авиационный институт

(Национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №0 по курсу «Машинное обучение»

Тема: Сбор и анализ данных

Студент: Харьков Павел Александрович

Группа: М80-406Б-19

Дата:

Оценка:

Москва, 2022

1. Постановка задачи

Необходимо найти или создать набор данных, проанализировать его, а затем привести его в формат для удобной работы с моделями.

2. Описание набора данных

Я выбрал набор данных, состоящий из 10-летних записей о погоде в городах Австралии и хотел бы построить такую модель, чтобы предсказывать, будет ли завтра дождь по состоянию сегодняшней погоды. Признаки этого датасета:

- Date – дата записи о погоде.
- Location – место записи о погоде.
- MinTemp – минимальная температура в этот день.
- MaxTemp – максимальная температура в этот день.
- Rainfall – количество осадков (в мм).
- Evaporation – количество испарений (в мм.
- Sunshine – количество солнечного света (в часах).
- WindGustDir – направление самого сильного ветра.
- WindGustSpeed – скорость самого сильного ветра.
- WindDir9am – направление ветра в 9 часов.
- WindDir3pm – направление ветра в 15 часов.
- WindSpeed9am – скорость ветра (в км/ч) в 9 часов.
- WindSpeed3pm – скорость ветра (в км/ч) в 15 часов.
- Humidity9am – влажность (в %) в 9 часов.
- Humidity3pm – влажность (в %) в 15 часов.
- Pressure9am – атмосферное давление (в hpa) в 9 часов.
- Pressure3pm – атмосферное давление (в hpa) в 15 часов.
- Cloud9am – количество неба, заполненное облаками (в oktas), в 9 часов.
- Cloud3pm – количество неба, заполненное облаками (в oktas), в 15 часов.
- Temp9am – температура (в C) в 9 часов.
- Temp3pm – температура (в C) в 15 часов.
- RainToday – был ли дождь сегодня.
- RainTomorrow – будет ли дождь завтра.

3. Ход работы

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
from sklearn import preprocessing
```

In [ ]:
```
pd.options.mode.chained_assignment = None
```

Сначала необходимо прочитать данные из csv и преобразовать дату из строкового формата в формат даты

In [ ]:
```
df = pd.read_csv('weatherAUS.csv')
df['Date'] = pd.to_datetime(df['Date'],format='%Y-%m-%d')
df
```

Out[ ]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	Wi
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	
...	...	...	...	...	...	...	...	...	
145455	2017-06-21	Uluru	2.8	23.4	0.0	NaN	NaN	E	
145456	2017-06-22	Uluru	3.6	25.3	0.0	NaN	NaN	NNW	
145457	2017-06-23	Uluru	5.4	26.9	0.0	NaN	NaN	N	
145458	2017-06-24	Uluru	7.8	27.0	0.0	NaN	NaN	SE	
145459	2017-06-25	Uluru	14.9	NaN	0.0	NaN	NaN	NaN	

145460 rows × 23 columns

Посмотрим, сколько нулевых значений в нашем датасете

In [ ]:
```
print(pd.isnull(df).sum())
```

```
Date                0
Location            0
MinTemp          1485
MaxTemp          1261
Rainfall         3261
Evaporation     62790
Sunshine        69835
WindGustDir     10326
WindGustSpeed   10263
WindDir9am      10566
WindDir3pm       4228
WindSpeed9am     1767
WindSpeed3pm     3062
Humidity9am      2654
Humidity3pm      4507
Pressure9am     15065
Pressure3pm     15028
Cloud9am        55888
Cloud3pm        59358
Temp9am          1767
Temp3pm          3609
RainToday        3261
RainTomorrow     3267
dtype: int64
```

Так как мы будем исследовать наш датасет относительно RainTomorrow, необходимо удалить все его неопределенные значения

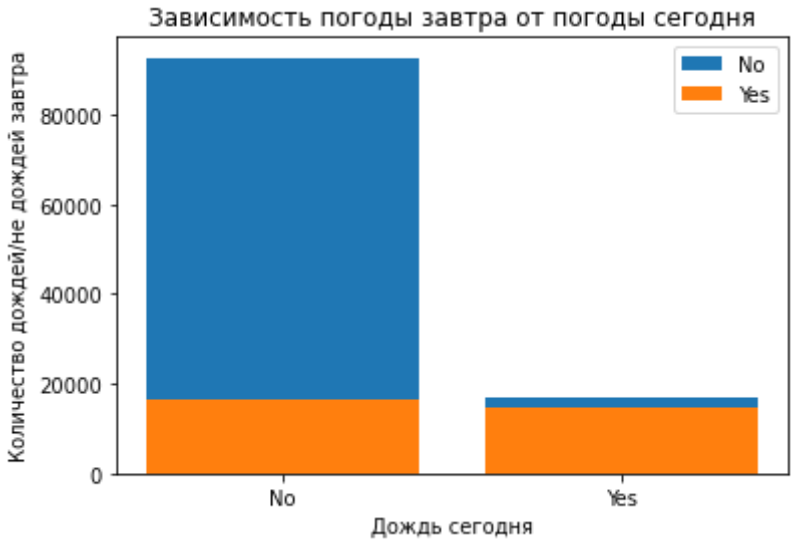
In [ ]:
```
df = df.dropna(subset=['RainToday'])
df = df.dropna(subset=['RainTomorrow'])
print(pd.isnull(df['RainTomorrow']).sum())
```

```
0
```

Визуализируем некоторые зависимости:

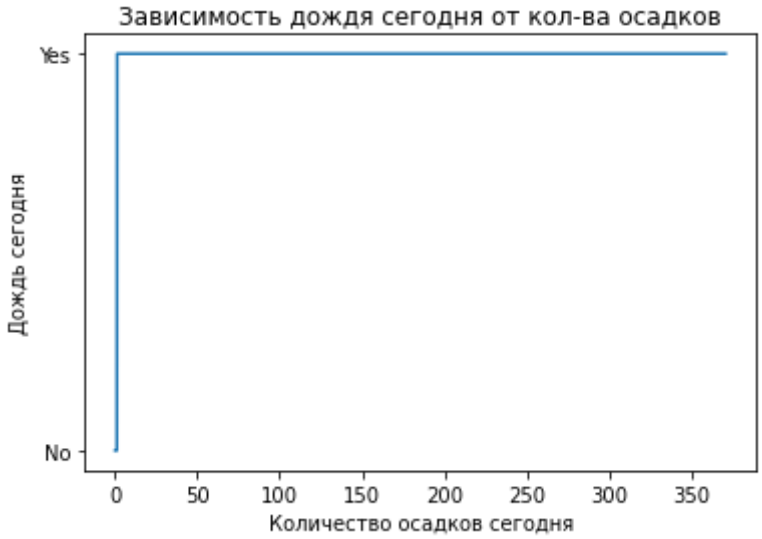
In [ ]:
```
plt.title("Зависимость погоды завтра от погоды сегодня")
plt.xlabel("Дождь сегодня")
plt.ylabel("Количество дождей/не дождей завтра")
plt.bar(['No', 'Yes'], [len(df[(df['RainToday'] == 'No') & (df['RainTomorrow'] ==
plt.bar(['No', 'Yes'], [len(df[(df['RainToday'] == 'No') & (df['RainTomorrow'] ==
plt.legend()
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x2103cbc92b0>
```



In [ ]:
```
plt.title("Зависимость дождя сегодня от кол-ва осадков")
plt.xlabel("Количество осадков сегодня")
plt.ylabel("Дождь сегодня")
df_tmp = df.sort_values(by=['Rainfall'])
plt.plot(df_tmp['Rainfall'], df_tmp['RainToday'])
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x2103d8d5ac0>]
```



Как видно, RainToday = 'Yes', если Rainfall >= 1, так что мы можем удалить столбец RainToday

In [ ]:
```
df = df.drop(['RainToday'], axis=1)
```

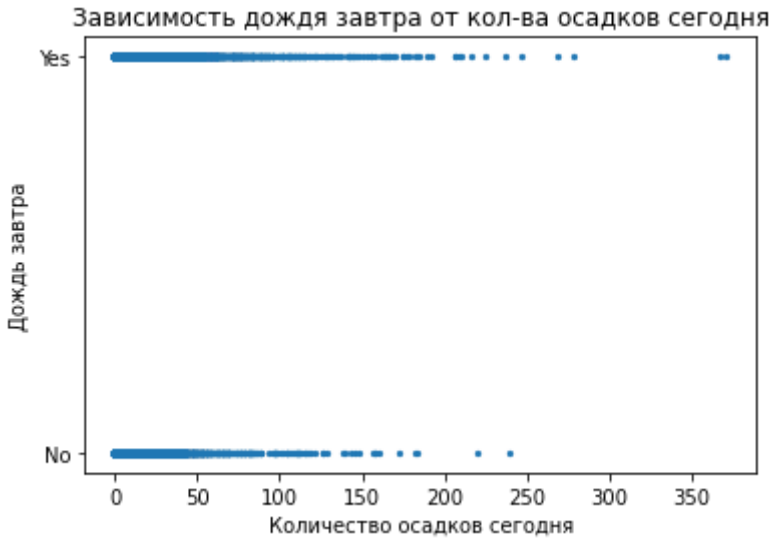
In [ ]:
```
plt.title("Зависимость кол-ва осадков сегодня от разницы температур")
plt.xlabel("Разница температур")
plt.ylabel("Количество осадков")
df_tmp = df.dropna(subset=['MinTemp', 'MaxTemp'])
df_tmp['TempDiff'] = abs(df_tmp['MaxTemp'].values - df_tmp['MinTemp'].values)
df_tmp = df_tmp.sort_values(by=['TempDiff'])
plt.scatter(df_tmp['TempDiff'], df_tmp['Rainfall'], s=5)
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x2103edd7fd0>
```

In [ ]:
```
plt.title("Зависимость дождя завтра от кол-ва осадков сегодня")
plt.xlabel("Количество осадков сегодня")
plt.ylabel("Дождь завтра")
```

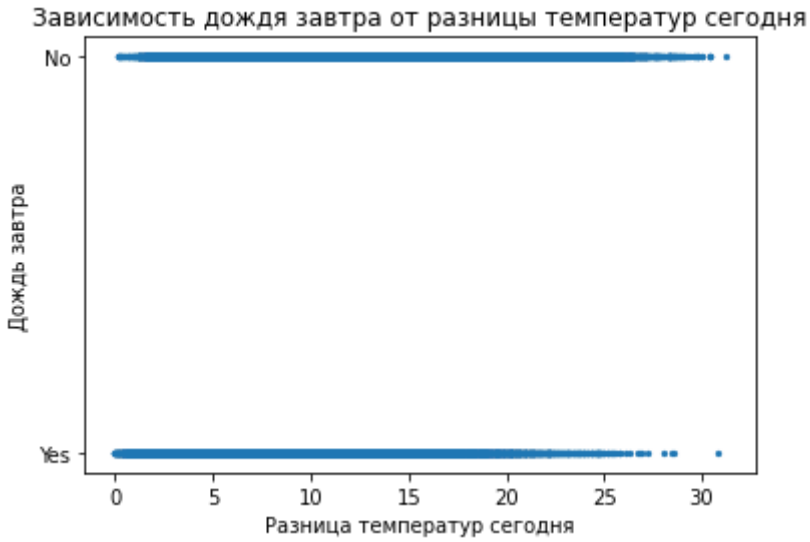
```
df_tmp = df.sort_values(by=['Rainfall'])
plt.scatter(df_tmp['Rainfall'], df_tmp['RainTomorrow'], s=5)
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x2103d973fd0>
```



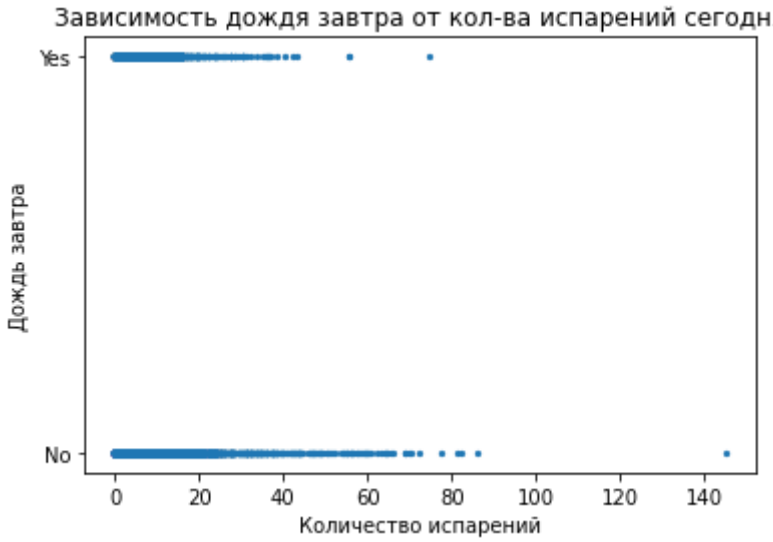
In [ ]:
```
plt.title("Зависимость дождя завтра от разницы температур сегодня")
plt.xlabel("Разница температур сегодня")
plt.ylabel("Дождь завтра")
df_tmp = df.dropna(subset=['MinTemp', 'MaxTemp'])
df_tmp['TempDiff'] = abs(df_tmp['MaxTemp'].values - df_tmp['MinTemp'].values)
df_tmp = df_tmp.sort_values(by=['TempDiff'])
plt.scatter(df_tmp['TempDiff'], df_tmp['RainTomorrow'], s=5)
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x2103d9afa90>
```



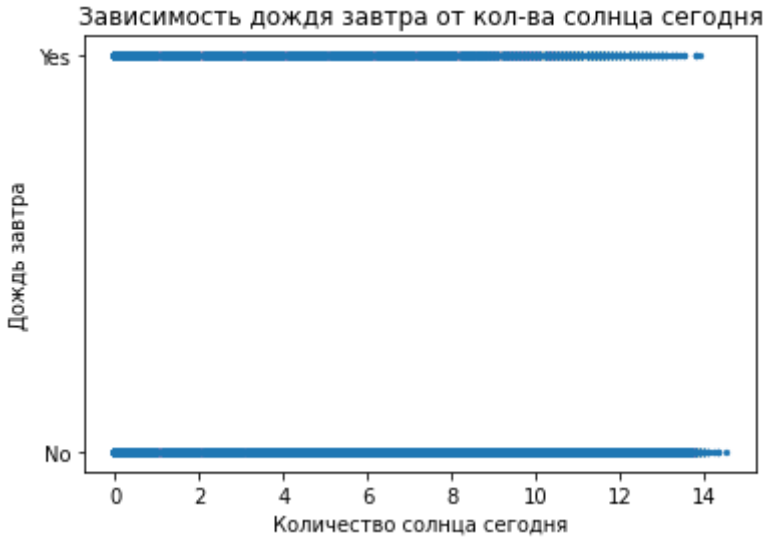
In [ ]:
```
plt.title("Зависимость дождя завтра от кол-ва испарений сегодня")
plt.xlabel("Количество испарений")
plt.ylabel("Дождь завтра")
df_tmp = df.dropna(subset=['Evaporation'])
df_tmp = df.sort_values(by=['Evaporation'])
plt.scatter(df_tmp['Evaporation'], df_tmp['RainTomorrow'], s=5)
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x2103ee0b520>
```



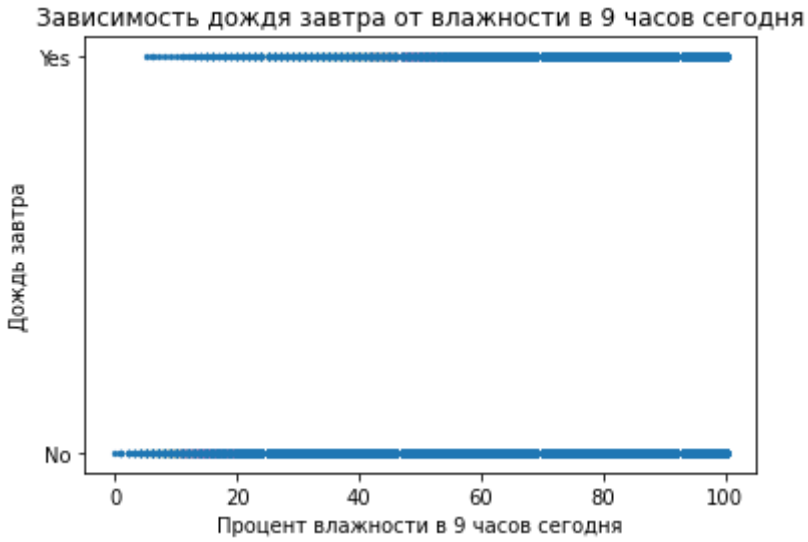
In [ ]:
```
plt.title("Зависимость дождя завтра от кол-ва солнца сегодня")
plt.xlabel("Количество солнца сегодня")
plt.ylabel("Дождь завтра")
df_tmp = df.dropna(subset=['Sunshine'])
df_tmp = df.sort_values(by=['Sunshine'])
plt.scatter(df_tmp['Sunshine'], df_tmp['RainTomorrow'], s=5)
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x2103ecb8fa0>
```



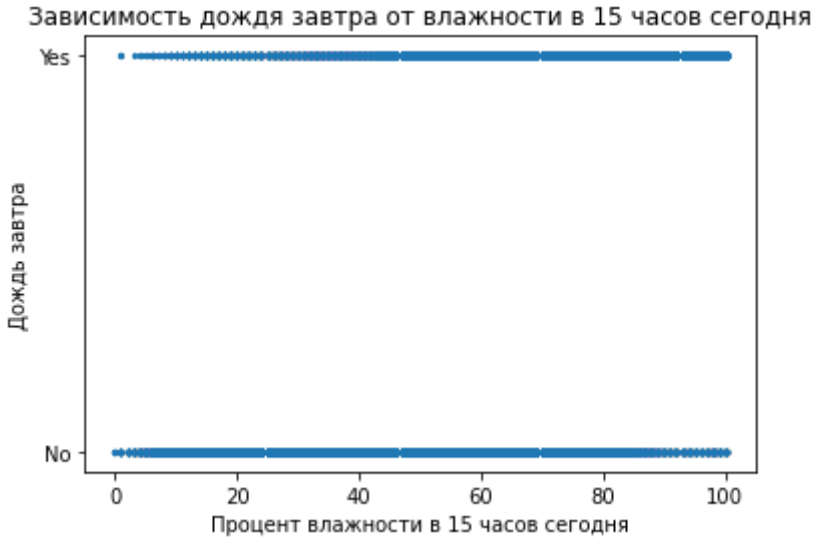
In [ ]:
```
plt.title("Зависимость дождя завтра от влажности в 9 часов сегодня")
plt.xlabel("Процент влажности в 9 часов сегодня")
plt.ylabel("Дождь завтра")
df_tmp = df.dropna(subset=['Humidity9am'])
df_tmp = df.sort_values(by=['Humidity9am'])
plt.scatter(df_tmp['Humidity9am'], df_tmp['RainTomorrow'], s=5)
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x2103ed15eb0>
```



In [ ]:
```
plt.title("Зависимость дождя завтра от влажности в 15 часов сегодня")
plt.xlabel("Процент влажности в 15 часов сегодня")
plt.ylabel("Дождь завтра")
df_tmp = df.dropna(subset=['Humidity3pm'])
df_tmp = df.sort_values(by=['Humidity3pm'])
plt.scatter(df_tmp['Humidity3pm'], df_tmp['RainTomorrow'], s=5)
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x2103ed6bf70>
```



Теперь необходимо заполнить нулевые значения в таблице, так как во многих столбцах их много, мы не можем их удалить. Заполнять нулевые значения мы будем не одним числом, а для каждого значения высчитывать среднее значение именно в этом городе в этом месяце. Благодаря этому мы получим более точные значения.

In [ ]:
```
for col in df.columns:
    if col == 'Date' or col == 'Location' or col == 'RainTomorrow':
        continue
    if col == 'WindGustDir' or col == 'WindDir9am' or col == 'WindDir3pm':
        s = df[col].value_counts().idxmax()
        df[col] = df.groupby(['Date', 'Location'])[col].apply(lambda x:x.fillna(x.
        continue
    m = df[col].mean()
    df[col] = df.groupby(['Date', 'Location'])[col].apply(lambda x:x.fillna(x.mean
```

Воспользуемся LabelEncoder для того, чтобы заменить значения в столбцах

RainTomorrow и Location

In [ ]:
```
rain_encoder = preprocessing.LabelEncoder()
df['RainTomorrow'] = rain_encoder.fit_transform(df['RainTomorrow'])
```

In [ ]:
```
loc_encoder = preprocessing.LabelEncoder()
df['Location'] = loc_encoder.fit_transform(df['Location'])
```

Воспользуемся OneHotEncoder для того, чтобы заменить значения в столбцах WindGustDir, WindDir9am и WindDir3pm. Однако делать мы будем это с помощью get_dummies, чтобы упростить работу с датасетом.

In [ ]:
```
df = pd.get_dummies(df,prefix=['WindGustDir'], columns = ['WindGustDir'], drop_firs
df = pd.get_dummies(df,prefix=['WindDir9am'], columns = ['WindDir9am'], drop_first=
df = pd.get_dummies(df,prefix=['WindDir3pm'], columns = ['WindDir3pm'], drop_first=
```

Удалим столбец даты, так как он наиболее уникальный:

In [ ]:
```
df = df.drop(['Date'], axis = 1)
```

А также сдвинем стобец RainTomorrow в правый конец для последующей работы с ним

In [ ]:
```
fd = df['RainTomorrow']
df_tmp = df.dropna(subset=['Location'])
df = df.drop(['RainTomorrow'], axis=1)
df['RainTomorrow'] = fd
```

Выведем полученный датасет

In [ ]:
```
df
```

Out[ ]:

	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindS
0	2	13.4	22.9	0.6	5.472516	7.63054	44.0	
1	2	7.4	25.1	0.0	5.472516	7.63054	44.0	
2	2	12.9	25.7	0.0	5.472516	7.63054	46.0	
3	2	9.2	28.0	0.0	5.472516	7.63054	24.0	
4	2	17.5	32.3	1.0	5.472516	7.63054	41.0	
...	...	...	...	...	...	...	...	
145454	41	3.5	21.8	0.0	5.472516	7.63054	31.0	
145455	41	2.8	23.4	0.0	5.472516	7.63054	31.0	
145456	41	3.6	25.3	0.0	5.472516	7.63054	22.0	
145457	41	5.4	26.9	0.0	5.472516	7.63054	37.0	
145458	41	7.8	27.0	0.0	5.472516	7.63054	28.0	

140787 rows × 63 columns

И сохраним его в новый файл

In [ ]:
```
df.to_csv('weatherAfter.csv')
```

In [ ]:
```

```

4. Выводы

Выполнив, данную лабораторную работу, я научился анализировать и изменять данные с помощью библиотек pandas, matplotlib и sklearn. Также я выявил такие зависимости в датасете, как:

- Чем больше разница температур, тем меньше осадков было сегодня.
- Чем больше осадков было сегодня, тем больше вероятность, что будет дождь завтра.
- Чем больше испарений было сегодня, тем меньше вероятность дождя завтра.
- Чем дольше было светило солнце сегодня, тем меньше вероятность дождя завтра.
- Чем больше процента влажности сегодня, тем больше вероятность дождя завтра.