МИНОБРНАУКИ РОССИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Южный федеральный университет»
Институт высоких технологий и пьезотехники

Кафедра прикладной информатики и инноватики

Направление: 09.03.03.01 "Прикладная информатика"

Большие данные

Отчёт по проекту

“ Сбор, предобработка и анализ информации, предсказание дождя на следующий день в Австралии”

Выполнили студенты 3 курса ВТ-09.03.03.01-о3 группы

_____ Ли К. Э.

подпись

_____ Джамалбеков А. Э.

подпись

Ростов-на-Дону – 2024

Содержание

1 Цели и задачи проекта 3
2 Ход работы 4
2.1 Импортирование библиотек 4
2.2 Загрузка данных 4
2.3 Визуализация данных 6
2.4 Предобработка данных 8
2.5 Кодирование 10
2.6 Разделение данных 11
2.7 Обучение модели и анализ предсказаний 11
3 Заключение **17**
4 Литература **18**

1 Цели и задачи проекта

Целью нашего проекта является обучение модели, которая будет удовлетворять следующим требованиям:

1. Модель должна предсказывать дождь на следующий день
2. Точность предсказания должна быть хорошей
3. Работа должна включать в себя полученные знания из курса

Для достижения целей проекта наша команда поставила перед собой некоторые задачи:

1. Предобработка и анализ данных
2. Изучение методов обучения и поиск наиболее подходящих для нашей модели
3. Повышение точности предсказания
4. Анализ предсказаний

Испокон веков люди пытались предсказать погодные изменения по приметам или с помощью гаданий. Современные методы прогнозирования значительно упростили жизнь. Например, вы когда-нибудь задумывались, стоит ли вам завтра брать с собой зонтик? С помощью этого проекта вы можете предсказать дождь на следующий день, используя машинное обучение.

2 Ход работы

2.1 Импортирование библиотек

Импортируем нужные библиотеки для дальнейшей работы.

In [23]:
```
from pyspark.ml import Pipeline
from IPython.display import display, HTML
from pyspark.ml.feature import StringIndexer, VectorAssembler
from pyspark.ml.classification import LogisticRegression, DecisionTreeClassifier, RandomForestClassifier, LinearSVC
from pyspark.ml.evaluation import BinaryClassificationEvaluator, MulticlassClassificationEvaluator
from pyspark.sql.types import StructType, StructField, StringType, DoubleType
from sklearn.metrics import roc_curve, auc, accuracy_score, f1_score
from pyspark.sql.functions import mean, when

import pyspark.sql.functions as F
import matplotlib.pyplot as plt
import missingno as msno
import warnings
import seaborn as sns
```

2.2 Загрузка данных

При импортировании датасета замечаем, что структура данных не соответствует действительности, поэтому необходимо задать ее вручную.

In [2]:
```
weatherDF = spark.read.option("sep", ",")\
    .option("header", True)\
    .csv("weatherAUS.csv")
```

In [3]:
```
weatherDF.printSchema()
weatherDF.show(5)

#dataframe shape
print((weatherDF.count(), len(weatherDF.columns)))
```

```
root
 |-- Date: string (nullable = true)
 |-- Location: string (nullable = true)
 |-- MinTemp: string (nullable = true)
 |-- MaxTemp: string (nullable = true)
 |-- Rainfall: string (nullable = true)
 |-- Evaporation: string (nullable = true)
 |-- Sunshine: string (nullable = true)
 |-- WindGustDir: string (nullable = true)
 |-- WindGustSpeed: string (nullable = true)
 |-- WindDir9am: string (nullable = true)
 |-- WindDir3pm: string (nullable = true)
 |-- WindSpeed9am: string (nullable = true)
 |-- WindSpeed3pm: string (nullable = true)
 |-- Humidity9am: string (nullable = true)
 |-- Humidity3pm: string (nullable = true)
 |-- Pressure9am: string (nullable = true)
 |-- Pressure3pm: string (nullable = true)
 |-- Cloud9am: string (nullable = true)
 |-- Cloud3pm: string (nullable = true)
 |-- Temp9am: string (nullable = true)
 |-- Temp3pm: string (nullable = true)
```

Задаем схематику датафрейма и подгружаем его.

In [4]:
```
schema = StructType([
    StructField("Date", StringType(), True),
    StructField("Location", StringType(), True),
    StructField("MinTemp", DoubleType(), True),
    StructField("MaxTemp", DoubleType(), True),
    StructField("Rainfall", DoubleType(), True),
    StructField("Evaporation", DoubleType(), True),
    StructField("Sunshine", DoubleType(), True),
    StructField("WindGustDir", StringType(), True),
    StructField("WindGustSpeed", DoubleType(), True),
    StructField("WindDir9am", StringType(), True),
    StructField("WindDir3pm", StringType(), True),
    StructField("WindSpeed9am", DoubleType(), True),
    StructField("WindSpeed3pm", DoubleType(), True),
    StructField("Humidity9am", DoubleType(), True),
    StructField("Humidity3pm", DoubleType(), True),
    StructField("Pressure9am", DoubleType(), True),
    StructField("Pressure3pm", DoubleType(), True),
    StructField("Cloud9am", DoubleType(), True),
    StructField("Cloud3pm", DoubleType(), True),
    StructField("Temp9am", DoubleType(), True),
    StructField("Temp3pm", DoubleType(), True),
    StructField("RainToday", StringType(), True),
    StructField("RainTomorrow", StringType(), True),
])
```

In [5]:
```
df = spark.read.option("sep", ",")\
    .option("header", True)\
    .csv("weatherAUS.csv", schema = schema)

df.printSchema()
df.show(5)
```

```
root
 |-- Date: string (nullable = true)
 |-- Location: string (nullable = true)
 |-- MinTemp: double (nullable = true)
 |-- MaxTemp: double (nullable = true)
 |-- Rainfall: double (nullable = true)
 |-- Evaporation: double (nullable = true)
 |-- Sunshine: double (nullable = true)
 |-- WindGustDir: string (nullable = true)
 |-- WindGustSpeed: double (nullable = true)
 |-- WindDir9am: string (nullable = true)
 |-- WindDir3pm: string (nullable = true)
 |-- WindSpeed9am: double (nullable = true)
 |-- WindSpeed3pm: double (nullable = true)
 |-- Humidity9am: double (nullable = true)
 |-- Humidity3pm: double (nullable = true)
 |-- Pressure9am: double (nullable = true)
 |-- Pressure3pm: double (nullable = true)
 |-- Cloud9am: double (nullable = true)
 |-- Cloud3pm: double (nullable = true)
 |-- Temp9am: double (nullable = true)
 |-- Temp3pm: double (nullable = true)
 |-- RainToday: string (nullable = true)
 |-- RainTomorrow: string (nullable = true)
```

Так выглядит наш датафрейм

```
+----------+--------+-------+-------+--------+-----------+--------+-----------+-------------+----------+----------+-----------+-----------+-----------+-----------
+----------+-----------+-------+-------+--------+-----------+--------+-----------+-------------+----------+----------+-----------+-----------+-----------+-----------
|      Date|Location|MinTemp|MaxTemp|Rainfall|Evaporation|Sunshine|WindGustDir|WindGustSpeed|WindDir9am|WindDir3pm|WindSpeed9am|WindSpeed3pm|Humidity9am|Humidity3p
m|Pressure9am|Pressure3pm|Cloud9am|Cloud3pm|Temp9am|Temp3pm|RainToday|RainTomorrow|
+----------+--------+-------+-------+--------+-----------+--------+-----------+-------------+----------+----------+-----------+-----------+-----------+-----------
+----------+-----------+-------+-------+--------+-----------+--------+-----------+-------------+----------+----------+-----------+-----------+-----------+-----------
|2008-12-01|  Albury|   13.4|   22.9|     0.6|       null|    null|          W|         44.0|         W|       WNW|       20.0|       24.0|       71.0|       22.
0|     1007.7|     1007.1|    8.0|    null|   16.9|   21.8|       No|          No|
|2008-12-02|  Albury|    7.4|   25.1|     0.0|       null|    null|        WNW|         44.0|       NNW|       WSW|        4.0|       22.0|       44.0|       25.
0|     1010.6|     1007.8|   null|    null|   17.2|   24.3|       No|          No|
|2008-12-03|  Albury|   12.9|   25.7|     0.0|       null|    null|        WSW|         46.0|         W|       WSW|       19.0|       26.0|       38.0|       30.
0|     1007.6|     1008.7|   null|     2.0|   21.0|   23.2|       No|          No|
|2008-12-04|  Albury|    9.2|   28.0|     0.0|       null|    null|         NE|         24.0|        SE|         E|       11.0|        9.0|       45.0|       16.
0|     1017.6|     1012.8|   null|    null|   18.1|   26.5|       No|          No|
|2008-12-05|  Albury|   17.5|   32.3|     1.0|       null|    null|          W|         41.0|       ENE|        NW|        7.0|       20.0|       82.0|       33.
0|     1010.8|     1006.0|    7.0|     8.0|   17.8|   29.7|       No|          No|
+----------+--------+-------+-------+--------+-----------+--------+-----------+-------------+----------+----------+-----------+-----------+-----------+-----------
+----------+-----------+-------+-------+--------+-----------+--------+-----------+-------------+----------+----------+-----------+-----------+-----------+-----------
only showing top 5 rows
```

Краткая сводка о столбцах

```
In [7]: for col in df.columns:
            df.select(col).summary().show()
        print((df.count(), len(df.columns)))
```

```
+-------+----------+
|summary|      Date|
+-------+----------+
|  count|    145460|
|   mean|      null|
| stddev|      null|
|    min|2007-11-01|
|    25%|      null|
|    50%|      null|
|    75%|      null|
|    max|2017-06-25|
+-------+----------+

+-------+--------+
|summary|Location|
+-------+--------+
|  count|  145460|
|   mean|    null|
| stddev|    null|
```

Датафрейм состоит из 145460 строк

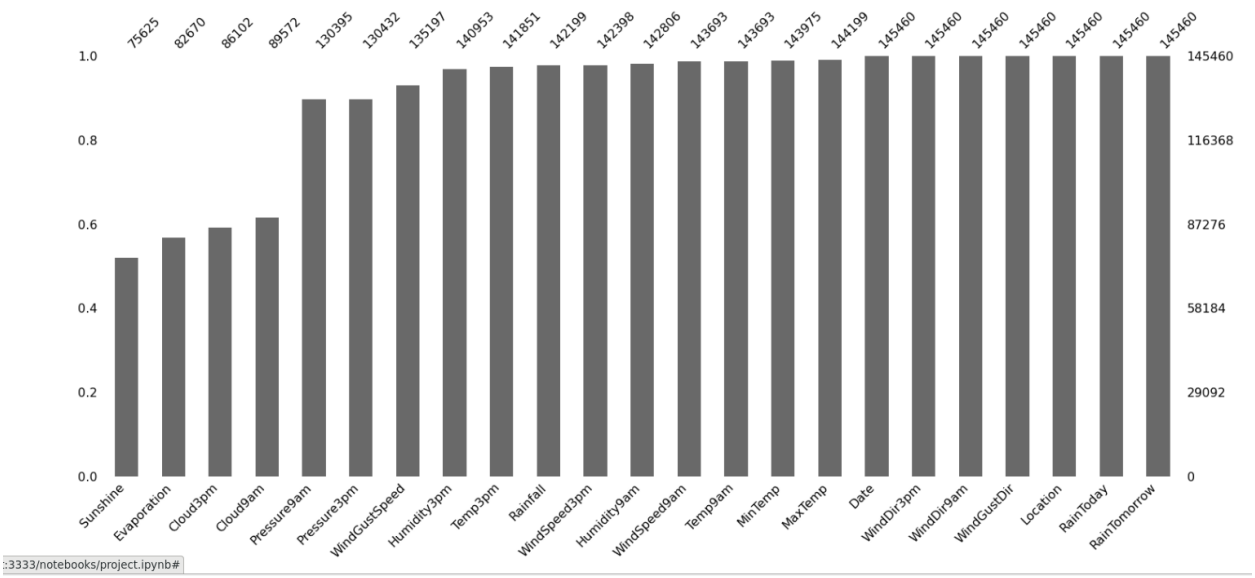
```
In [39]: print(df.count())
         145460
```

2.3 Визуализация данных

Диаграмма количества строк в каждом столбце.

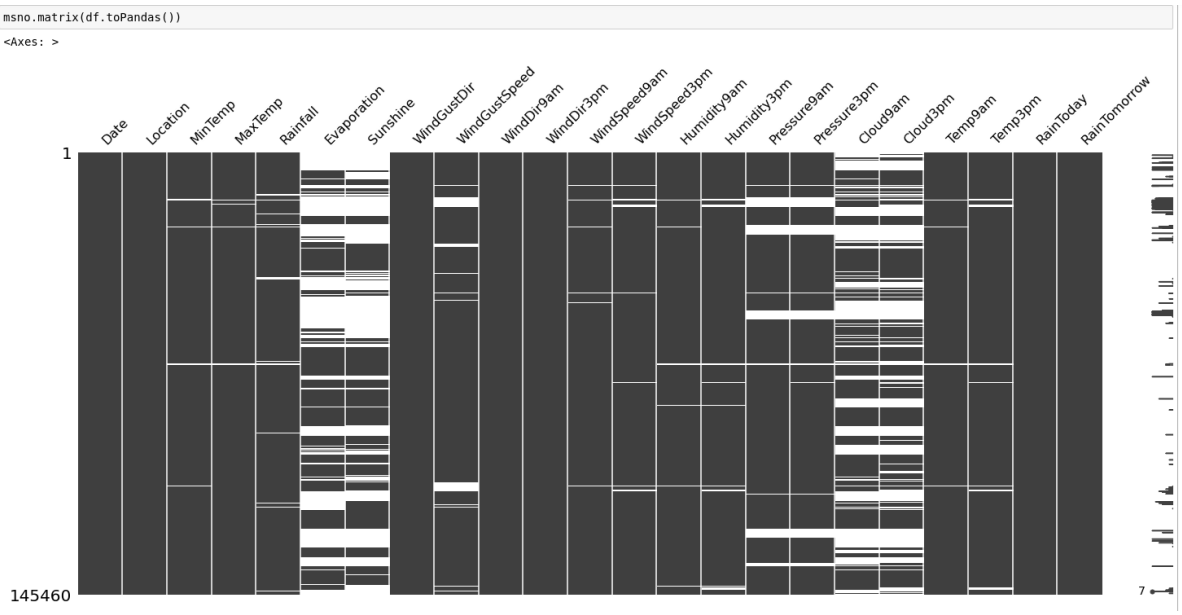
Можно заметить большое количество отсутствующих данных в столбцах Sunshine, Evaporation, Cloud9am, Cloud3pm

```
In [8]: msno.bar(df.toPandas(), sort='ascending')
Out[8]: <Axes: >
```



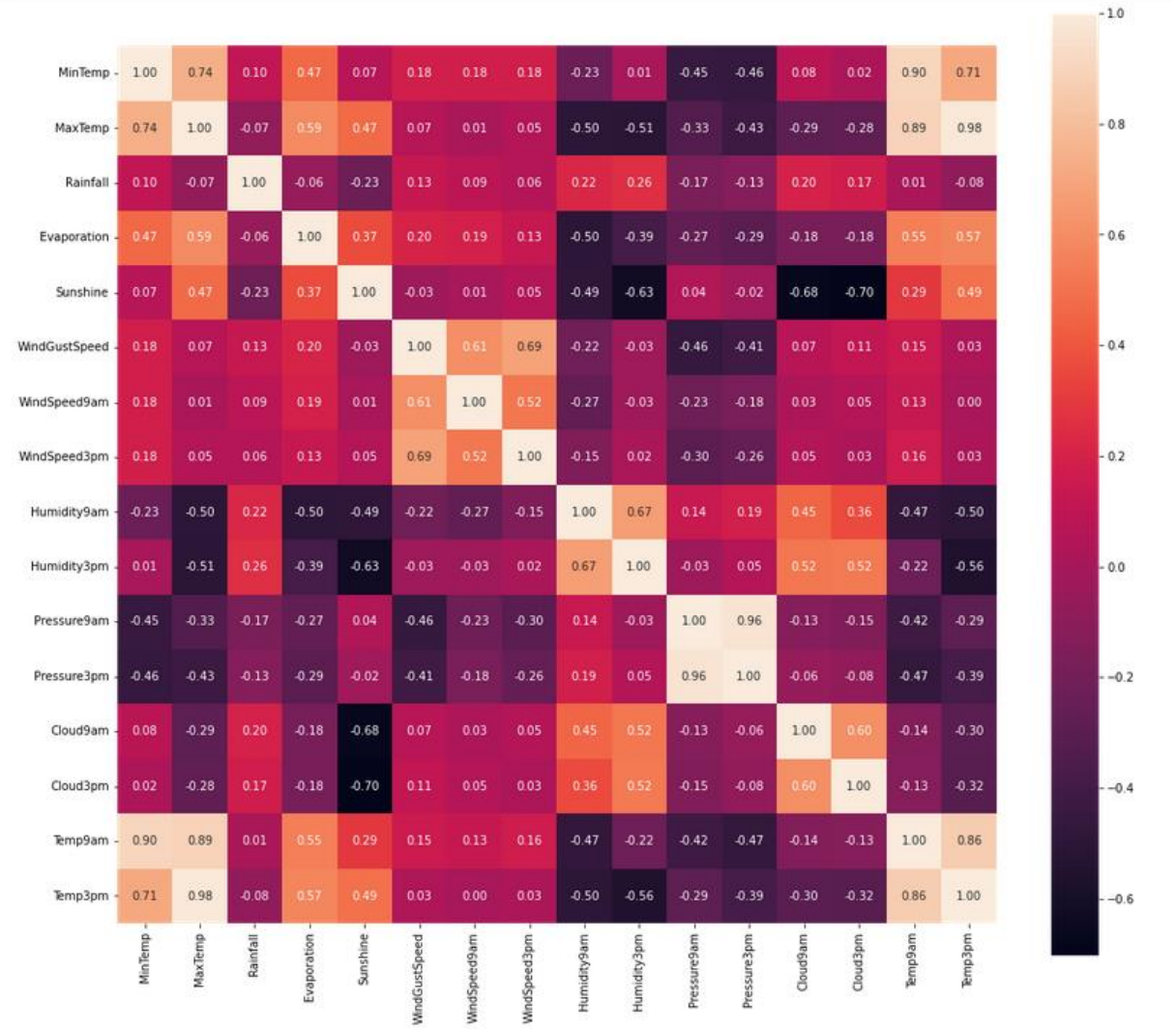
Заполненность данных построчно.

```
In [9]: msno.matrix(df.toPandas())
Out[9]: <Axes: >
```



Heatmap покажет зависимость одного признака от другого. Индекс корреляции варьируется от -1 до 1. Если значение близится к -1, то признаки обратно пропорциональны друг другу, к 1, то прямо пропорциональны, 0 – не зависят.

```
In [10]: plt.figure(figsize=(17,15))
         ax = sns.heatmap(df.toPandas().corr(), square=True, annot=True, fmt='.2f')
         ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
         plt.show()
```



2.4 Предобработка данных

Вывод количества null и NaN значений в каждом столбце.

In [11]:
```
Dict_Null = {col:df.filter(df[col].isNull()).count() for col in df.columns}
Dict_Null
```

Out[11]:
```
{'Date': 0,
 'Location': 0,
 'MinTemp': 1485,
 'MaxTemp': 1261,
 'Rainfall': 3261,
 'Evaporation': 62790,
 'Sunshine': 69835,
 'WindGustDir': 0,
 'WindGustSpeed': 10263,
 'WindDir9am': 0,
 'WindDir3pm': 0,
 'WindSpeed9am': 1767,
 'WindSpeed3pm': 3062,
 'Humidity9am': 2654,
 'Humidity3pm': 4507,
 'Pressure9am': 15065,
 'Pressure3pm': 15028,
 'Cloud9am': 55888,
 'Cloud3pm': 59358,
 'Temp9am': 1767,
 'Temp3pm': 3609,
 'RainToday': 0,
 'RainTomorrow': 0}
```

Null значения будут заменены на средние в этом столбце. Это позволит не терять необходимые данные для предсказания, поскольку пробелы в датах не позволят нам покрывать все дни. Хотя большое количество одинаковых данных может привести к плохому обучению из-за однородности признака.

In [40]:
```
#Filling the missing values for continuous variables with mean
for col in df.columns:
    # Подсчитываем среднее значение столбца
    mean_value = df.select(mean(df[col])).collect()[0][0]
    # Заменяем null значения на среднее значение столбца
    df = df.withColumn(col, when(df[col].isNull(), mean_value).otherwise(df[col]))

#deleting null values
print(df.count())
```

```
145460
```

In [29]:
```
df.select('Evaporation', 'Sunshine', 'Cloud9am', 'Cloud3pm').show()
```

```
+-----------------+-----------------+-----------------+-----------------+
|      Evaporation|         Sunshine|          Cloud9am|         Cloud3pm|
+-----------------+-----------------+-----------------+-----------------+
|5.468231522922461|7.611177520661216|              8.0|4.509930082924903|
|5.468231522922461|7.611177520661216|4.447461260215245|4.509930082924903|
|5.468231522922461|7.611177520661216|4.447461260215245|              2.0|
|5.468231522922461|7.611177520661216|4.447461260215245|4.509930082924903|
|5.468231522922461|7.611177520661216|              7.0|              8.0|
|5.468231522922461|7.611177520661216|4.447461260215245|4.509930082924903|
|5.468231522922461|7.611177520661216|              1.0|4.509930082924903|
|5.468231522922461|7.611177520661216|4.447461260215245|4.509930082924903|
|5.468231522922461|7.611177520661216|4.447461260215245|4.509930082924903|
|5.468231522922461|7.611177520661216|4.447461260215245|4.509930082924903|
|5.468231522922461|7.611177520661216|4.447461260215245|4.509930082924903|
|5.468231522922461|7.611177520661216|              8.0|              8.0|
|5.468231522922461|7.611177520661216|              8.0|              8.0|
|5.468231522922461|7.611177520661216|4.447461260215245|              7.0|
|5.468231522922461|7.611177520661216|              8.0|              1.0|
|5.468231522922461|7.611177520661216|              8.0|              1.0|
|5.468231522922461|7.611177520661216|4.447461260215245|              2.0|
|5.468231522922461|7.611177520661216|4.447461260215245|4.509930082924903|
|5.468231522922461|7.611177520661216|4.447461260215245|4.509930082924903|
|5.468231522922461|7.611177520661216|4.447461260215245|              1.0|
+-----------------+-----------------+-----------------+-----------------+
only showing top 20 rows
```

```
In [14]: Dict_Null = {col:df.filter(df[col].isNull()).count() for col in df.columns}
         Dict_Null
```

```
Out[14]: {'Date': 0,
          'Location': 0,
          'MinTemp': 0,
          'MaxTemp': 0,
          'Rainfall': 0,
          'Evaporation': 0,
          'Sunshine': 0,
          'WindGustDir': 0,
          'WindGustSpeed': 0,
          'WindDir9am': 0,
          'WindDir3pm': 0,
          'WindSpeed9am': 0,
          'WindSpeed3pm': 0,
          'Humidity9am': 0,
          'Humidity3pm': 0,
          'Pressure9am': 0,
          'Pressure3pm': 0,
          'Cloud9am': 0,
          'Cloud3pm': 0,
          'Temp9am': 0,
          'Temp3pm': 0,
          'RainToday': 0,
          'RainTomorrow': 0}
```

RainToday и RainTomorrow должен иметь 2 категории – yes/no, однако есть и NA значения, которые указывают на отсутствие данных. Поскольку столбец String типа эти пробелы не были обнаружены в визуализации данных. Удаляем эти строки. Потеряли около пяти тысяч строк.

```
In [41]: df = df.filter((F.col("RainToday") != "NA") & (F.col("RainTomorrow") != "NA") )
         print(df.count())
```

```
140787
```

2.5 Кодирование

Датафрейм имеет как числовые, так и категориальные признаки. Необходимо перевести все в числовой тип для дальнейшей работы с моделью.

```
In [16]: #Encoding
encoder = StringIndexer(inputCols=['RainToday', 'RainTomorrow', 'Location', 'WindDir9am', 'WindDir3pm', 'WindGustDir'],\
                        outputCols = ['RainTodayBin', 'RainTomorrowBin','LocationBin', 'WindDir9amBin', 'WindDir3pmBin', 'WindGustDirBin'])
df = encoder.fit(df).transform(df)
df = df.drop('RainToday', 'RainTomorrow', 'Location', 'WindDir9am', 'WindDir3pm', 'WindGustDir')
```

```
In [17]: df.select('RainTodayBin', 'RainTomorrowBin','LocationBin', 'WindDir9amBin', 'WindDir3pmBin', 'WindGustDirBin').show()
```

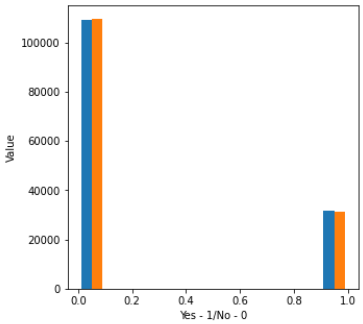
```
+------------+---------------+-----------+-------------+-------------+--------------+
|RainTodayBin|RainTomorrowBin|LocationBin|WindDir9amBin|WindDir3pmBin|WindGustDirBin|
+------------+---------------+-----------+-------------+-------------+--------------+
|         0.0|            0.0|       20.0|          7.0|          7.0|           0.0|
|         0.0|            0.0|       20.0|         10.0|          3.0|          10.0|
|         0.0|            0.0|       20.0|          7.0|          3.0|           7.0|
|         0.0|            0.0|       20.0|          2.0|         10.0|          14.0|
|         0.0|            0.0|       20.0|         11.0|          8.0|           0.0|
|         0.0|            0.0|       20.0|          7.0|          1.0|          10.0|
|         0.0|            0.0|       20.0|          8.0|          1.0|           0.0|
|         0.0|            0.0|       20.0|          4.0|          1.0|           0.0|
|         0.0|            1.0|       20.0|          2.0|          8.0|          15.0|
|         1.0|            0.0|       20.0|          6.0|          5.0|           0.0|
|         0.0|            1.0|       20.0|          4.0|          9.0|           4.0|
|         1.0|            1.0|       20.0|         13.0|         14.0|          16.0|
|         1.0|            1.0|       20.0|         10.0|         13.0|           0.0|
|         1.0|            0.0|       20.0|          7.0|         12.0|           8.0|
|         0.0|            1.0|       20.0|         14.0|         10.0|          12.0|
|         1.0|            1.0|       20.0|          0.0|          7.0|           0.0|
|         1.0|            0.0|       20.0|         16.0|          4.0|           5.0|
|         0.0|            0.0|       20.0|          2.0|         13.0|           5.0|
|         0.0|            0.0|       20.0|          2.0|          0.0|           6.0|
|         0.0|            0.0|       20.0|         13.0|          6.0|          14.0|
+------------+---------------+-----------+-------------+-------------+--------------+
only showing top 20 rows
```

Просмотр данных о дожде. Представление его количества на диаграмме.

In [43]:
```
print("Rain tomorrow",df.select("*").where("RainTomorrowBin == 1").count(), df.select("*").where("RainTomorrowBin == 0").count())
print("Rain today",df.select("*").where("RainTodayBin == 1").count(), df.select("*").where("RainTodayBin == 0").count())


#Preparing to build hist
rainToday = [val.RainTodayBin for val in df.select('RainTodayBin').collect()]
rainTomorrow = [val.RainTomorrowBin for val in df.select('RainTomorrowBin').collect()]
mas = [rainToday, rainTomorrow]

#Histogram
plt.hist(mas)
plt.xlabel('Yes - 1/No - 0')
plt.ylabel('Value')
```

```
Rain tomorrow 31201 109586
Rain today 31455 109332
```

Out[43]:
```
Text(0, 0.5, 'Value')
```



2.6 Разделение данных

Для обучения модели создан вектор признаков, который включает себя все столбцы, кроме RainTomorrowBin, поскольку он используется для предсказания.

In [44]:
```
featureCols = ['MinTemp','LocationBin', 'WindDir9amBin', 'WindDir3pmBin', 'WindGustDirBin',
    'MaxTemp',
    'Rainfall',
    'Evaporation',
    'Sunshine',
    'WindGustSpeed',
    'WindSpeed9am',
    'WindSpeed3pm',
    'Humidity9am',
    'Humidity3pm',
    'Pressure9am',
    'Pressure3pm',
    'Cloud9am',
    'Cloud3pm',
    'Temp9am',
    'Temp3pm',
 'RainTodayBin']
assembler = VectorAssembler(inputCols=featureCols, outputCol='features')

#Splitting the data into train and test
train_data, test_data = df.randomSplit([0.7, 0.3])
```

Разделение данных на тренировочные и тестовые будет происходить случайным образом. Отношение 70% на 30%.

2.7 Обучение модели и анализ предсказаний

В качестве методов обучения выбраны DecisionTree, RandomForest, LinearSVC.

Создание метода для отрисовки ROC кривой

```
def calc_ROC(y_test, y_predict):
    fpr, tpr, treshold = roc_curve(y_test, y_pred)
    roc_auc = auc(fpr, tpr)
    plt.rcParams.update({'figure.figsize': (5, 5)})
    plt.plot(fpr, tpr, color='red',label='ROC кривая (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
    plt.xlim([-0.005, 1.0])
    plt.ylim([0.0, 1.005])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC-кривая')
    plt.legend(loc="lower right")
    plt.show()
```

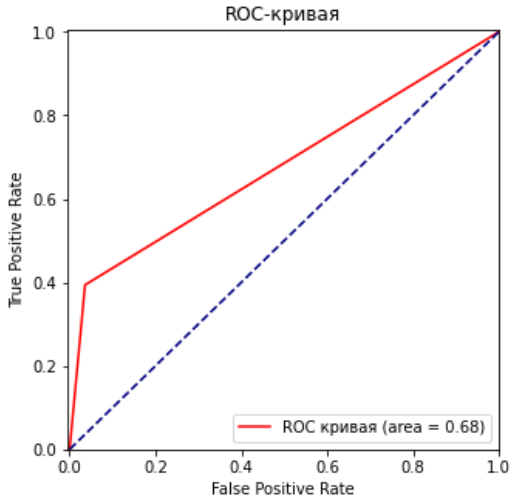
Метод решающего дерева основан на определении, в какую из дочерних вершин нужно поместить рассматриваемый объект. Строится дерево из признаков, и, в зависимости от тех или иных значений, получаем результат будет дождь или нет

```
In [45]: #DECISION TREE
df_result_dt = df

dt = DecisionTreeClassifier(labelCol="RainTomorrowBin", featuresCol="features", maxBins = 49)
pipe = Pipeline(stages=[assembler, dt])

#Fitting
fit_model = pipe.fit(train_data)

#Storing the results
df_result_dt = fit_model.transform(test_data)

df_result_dt.select("Date","LocationBin", "RainTodayBin","RainTomorrowBin","rawPrediction","probability","prediction").show()


evaluator = MulticlassClassificationEvaluator(
    labelCol="RainTomorrowBin", predictionCol="prediction", metricName="accuracy")
accuracy_dt = evaluator.evaluate(df_result_dt)
print(accuracy_dt*100)

y_test = df_result_dt.select("RainTomorrowBin").collect()
y_pred = df_result_dt.select("prediction").collect()

calc_ROC(y_test, y_pred)
print("F1 score:",f1_score(y_test, y_pred,)*100, "%")
```

Таблица и точность предсказания.

```
+----------+-----------+------------+---------------+---------------+--------------------+----------+
|      Date|LocationBin|RainTodayBin|RainTomorrowBin|   rawPrediction|         probability|prediction|
+----------+-----------+------------+---------------+---------------+--------------------+----------+
|2008-02-01|        1.0|         1.0|            1.0|   [439.0,651.0]|[0.40275229357798...|       1.0|
|2008-02-10|        1.0|         1.0|            0.0|[23543.0,5426.0]|[0.81269633056025...|       0.0|
|2008-02-11|        1.0|         0.0|            0.0|[23543.0,5426.0]|[0.81269633056025...|       0.0|
|2008-02-17|        1.0|         0.0|            0.0|[23543.0,5426.0]|[0.81269633056025...|       0.0|
|2008-02-24|        1.0|         0.0|            0.0|[43756.0,3632.0]|[0.92335612391322...|       0.0|
|2008-02-26|        1.0|         0.0|            1.0|[23543.0,5426.0]|[0.81269633056025...|       0.0|
|2008-02-28|        1.0|         1.0|            1.0|  [788.0,4654.0]|[0.14479970599044...|       1.0|
|2008-03-03|        1.0|         0.0|            0.0|[23543.0,5426.0]|[0.81269633056025...|       0.0|
|2008-03-08|        1.0|         1.0|            0.0|[23543.0,5426.0]|[0.81269633056025...|       0.0|
|2008-03-11|        1.0|         0.0|            0.0|[23543.0,5426.0]|[0.81269633056025...|       0.0|
|2008-03-13|        1.0|         0.0|            0.0|[23543.0,5426.0]|[0.81269633056025...|       0.0|
|2008-03-14|        1.0|         0.0|            0.0|[23543.0,5426.0]|[0.81269633056025...|       0.0|
|2008-03-23|        1.0|         0.0|            1.0|[23543.0,5426.0]|[0.81269633056025...|       0.0|
|2008-03-24|        1.0|         1.0|            0.0|   [701.0,555.0]|[0.55812101910828...|       0.0|
|2008-03-25|        1.0|         0.0|            1.0|[23543.0,5426.0]|[0.81269633056025...|       0.0|
|2008-03-26|        1.0|         1.0|            0.0|[43756.0,3632.0]|[0.92335612391322...|       0.0|
|2008-03-27|        1.0|         0.0|            0.0|[43756.0,3632.0]|[0.92335612391322...|       0.0|
|2008-03-28|        1.0|         0.0|            0.0|[43756.0,3632.0]|[0.92335612391322...|       0.0|
|2008-03-29|        1.0|         0.0|            0.0|[23543.0,5426.0]|[0.81269633056025...|       0.0|
|2008-04-01|        1.0|         0.0|            0.0|[23543.0,5426.0]|[0.81269633056025...|       0.0|
+----------+-----------+------------+---------------+---------------+--------------------+----------+
only showing top 20 rows

83.61640972040763
```

ROC кривая и F1 score.




```
F1 score: 51.89370161121574 %
```

Метод случайного леса строит множество деревьев решений во время обучения и выводит класс в режиме классификации или среднее предсказание в режиме регрессий. Каждое дерево в случайном лесу обучается на различном подмножестве данных, а случайность, добавленная во время обучения, помогает улучшить общие показатели модели и ее обощающие способности.

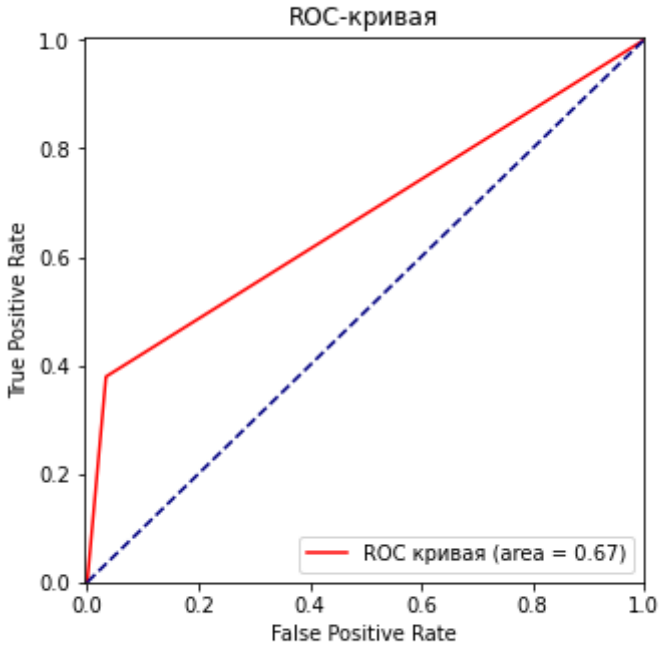
In [25]:
```
#RANDOM FOREST
df_result_rf = df

rf = RandomForestClassifier(labelCol="RainTomorrowBin", featuresCol="features", numTrees=10, maxBins = 49)
pipe = Pipeline(stages=[assembler, rf])

#Fitting
fit_model = pipe.fit(train_data)

#Storing the results
df_result_rf = fit_model.transform(test_data)

df_result_rf.select("Date","LocationBin", "RainTodayBin","RainTomorrowBin","rawPrediction","probability","prediction").show(50)

#Accuracy
evaluator = MulticlassClassificationEvaluator(
    labelCol="RainTomorrowBin", predictionCol="prediction", metricName="accuracy")
accuracy_rf = evaluator.evaluate(df_result_rf)
print(accuracy_rf*100)

y_test = df_result_rf.select("RainTomorrowBin").collect()
y_pred = df_result_rf.select("prediction").collect()

calc_ROC(y_test, y_pred)
print("F1 score:",f1_score(y_test, y_pred,)*100, "%")
```

Таблица и точность предсказания.

Date	LocationBin	RainTodayBin	RainTomorrowBin	rawPrediction	probability	prediction
2008-02-02	1.0	1.0	1.0	[4.92200133773597...	[0.49220013377359...	1.0
2008-02-03	1.0	1.0	1.0	[1.65582223434903...	[0.16558222343490...	1.0
2008-02-04	1.0	1.0	1.0	[1.77288046608767...	[0.17728804660876...	1.0
2008-02-10	1.0	1.0	0.0	[8.02067403303658...	[0.80206740330365...	0.0
2008-02-11	1.0	0.0	0.0	[8.27615942095491...	[0.82761594209549...	0.0
2008-02-16	1.0	0.0	0.0	[8.34424927078439...	[0.83442492707843...	0.0
2008-02-20	1.0	0.0	0.0	[9.01066345240779...	[0.90106634524077...	0.0
2008-02-29	1.0	1.0	0.0	[8.27735693391029...	[0.82773569339102...	0.0
2008-03-01	1.0	0.0	1.0	[9.05114835365764...	[0.90511483536576...	0.0
2008-03-02	1.0	1.0	0.0	[7.89502392531392...	[0.78950239253139...	0.0
2008-03-07	1.0	0.0	1.0	[7.00881781078911...	[0.70088178107891...	0.0
2008-03-08	1.0	1.0	0.0	[6.27253510415454...	[0.62725351041545...	0.0
2008-03-11	1.0	0.0	0.0	[9.01066345240779...	[0.90106634524077...	0.0
2008-03-12	1.0	0.0	0.0	[9.01066345240779...	[0.90106634524077...	0.0
2008-03-19	1.0	0.0	0.0	[8.29950019429208...	[0.82995001942920...	0.0
2008-03-22	1.0	1.0	0.0	[7.44090829933289...	[0.74409082993328...	0.0
2008-03-23	1.0	0.0	1.0	[8.40451223478648...	[0.84045122347864...	0.0
2008-03-25	1.0	0.0	1.0	[6.68746055706658...	[0.66874605570665...	0.0
2008-03-28	1.0	0.0	0.0	[9.05114835365764...	[0.90511483536576...	0.0
2008-04-02	1.0	0.0	0.0	[8.98305850382816...	[0.89830585038281...	0.0
2008-04-06	1.0	0.0	1.0	[8.40451223478648...	[0.84045122347864...	0.0
2008-04-08	1.0	1.0	1.0	[4.79110217376719...	[0.47911021737671...	1.0

ROC кривая и F1 score

```
83.45250255362615
```




```
F1 score: 50.693657984144956 %
```

LinearSVC - алгоритм классификации, в основе которого лежит определяемая разделяющая гиперплоскость (линия, прямая, многомерные плоскости).

При заданных тренировочных данных алгоритм находит такую гиперплоскость, которая разделяет данные, принадлежащие разным классам, самым оптимальным способом. В двухмерном пространстве гиперплоскостью служит прямая линия.

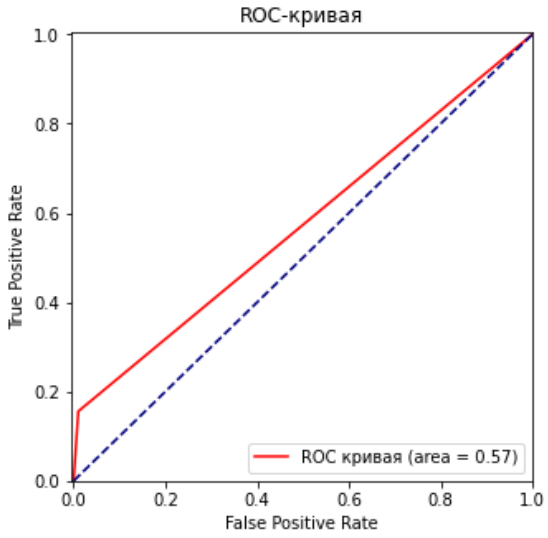
In [47]:
```
#SVC
df_result_svc = df

lsvc = LinearSVC(labelCol="RainTomorrowBin", featuresCol="features",maxIter=100, regParam=0.1)
pipe = Pipeline(stages=[assembler, lsvc])

#Fit
fit_model = pipe.fit(train_data)

#Storing the results
df_result_svc = fit_model.transform(test_data)

df_result_svc.select("Date","LocationBin", "RainTodayBin","RainTomorrowBin","rawPrediction","prediction").show(50)
#Accuracy
evaluator = MulticlassClassificationEvaluator(
    labelCol="RainTomorrowBin", predictionCol="prediction", metricName="accuracy")
accuracy_svc = evaluator.evaluate(df_result_svc)
print(accuracy_svc*100)

y_test = df_result_svc.select("RainTomorrowBin").collect()
y_pred = df_result_svc.select("prediction").collect()

calc_ROC(y_test, y_pred)
print("F1 score :",f1_score(y_test, y_pred,)*100, "%")
```

Таблица предсказаний

Date	LocationBin	RainTodayBin	RainTomorrowBin	rawPrediction	prediction
2008-02-01	1.0	1.0	1.0	[-0.4798361411210...	1.0
2008-02-10	1.0	1.0	0.0	[0.98860324565300...	0.0
2008-02-11	1.0	0.0	0.0	[1.02960276755638...	0.0
2008-02-17	1.0	0.0	0.0	[0.54702529610036...	0.0
2008-02-24	1.0	0.0	0.0	[1.54950153374232...	0.0
2008-02-26	1.0	0.0	1.0	[0.90379450024942...	0.0
2008-02-28	1.0	1.0	1.0	[-0.1844214977683...	1.0
2008-03-03	1.0	0.0	0.0	[1.30101046266245...	0.0
2008-03-08	1.0	1.0	0.0	[0.43943591717093...	0.0
2008-03-11	1.0	0.0	0.0	[1.27041063754803...	0.0
2008-03-13	1.0	0.0	0.0	[1.01989706393783...	0.0
2008-03-14	1.0	0.0	0.0	[1.17890613669859...	0.0
2008-03-23	1.0	0.0	1.0	[0.86941088947717...	0.0
2008-03-24	1.0	1.0	0.0	[0.30754179808812...	0.0
2008-03-25	1.0	0.0	1.0	[0.53092151457853...	0.0
2008-03-26	1.0	1.0	0.0	[0.87838281139276...	0.0
2008-03-27	1.0	0.0	0.0	[1.46873587984845...	0.0
2008-03-28	1.0	0.0	0.0	[1.15170657407024...	0.0
2008-03-29	1.0	0.0	0.0	[1.02440787954967...	0.0
2008-04-01	1.0	0.0	0.0	[1.32486400510276...	0.0
2008-04-03	1.0	0.0	0.0	[1.62743987224117...	0.0
2008-04-05	1.0	0.0	0.0	[1.39649284817279...	0.0
2008-04-07	1.0	1.0	1.0	[0.24880157309502...	0.0
2008-04-13	1.0	0.0	1.0	[0.67254012943364...	0.0
2008-04-20	1.0	1.0	1.0	[0.21090394231141...	0.0
2008-04-21	1.0	1.0	1.0	[-0.0782288689855...	1.0
2008-04-23	1.0	1.0	1.0	[-0.3260576208255...	1.0
2008-04-24	1.0	1.0	1.0	[-0.1746320254661...	1.0

Хотя и точность неплохая, ROC кривая и F1 score дают понять, что большое количество данных предсказаны ошибочно.

```
80.26937786540608
```




```
F1 score : 26.155761024182073 %
```

3 Заключение

В ходе работы построена модель предсказания данных о дожде на следующий день. Было использовано 3 метода обучения: решающее дерево, случайный лес и метод опорных векторов. Лучшие предсказания точности были с использованием решающего дерева. Точность составила 83.6 %, что является хорошим результатом в сочетании с результатами F1 score и ROC score. Наихудшее обучение было с методом опорных векторов, хотя точность и составляла 80%, F1 score был очень низким, что указывает на плохую точность и классификацию данных.

4 Литература

1. Датасет [Электронный ресурс]

URL: https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package/data

2. Пример анализа и обработки данных [Электронный ресурс]

URL: https://www.kaggle.com/code/fahadmehfoooz/rain-prediction-with-90-65-accuracy#3.-Training-The-Models

3.Обучение модели [Электронный ресурс]

URL: https://www.kaggle.com/code/prashant111/logistic-regression-classifier-tutorial#8.-Declare-feature-vector-and-target-variable-

4. Методы обучения [Электронный ресурс]

URL: https://habr.com/ru/companies/ods/articles/322534/