Отчет по проекту

«Предсказание средней температуры»

Команда

Гончарова Мария Сергеевна; Парамонова Екатерина Юрьевна

1. Постановка задачи

Данный кейс предлагает классическую задачу предсказания средней температуры с использованием методов регрессии. Это предполагает анализ исторических метеорологических данных для создания модели, которая сможет предсказывать среднюю температуру на основе различных факторов.

Для достижения этой цели необходимо использовать данные о погоде, собранные в течение нескольких лет. Эти данные включают различные метеорологические показатели, такие как облачный покров, солнечный свет, глобальное излучение, максимальная и минимальная температура, атмосферные осадки, давление и глубина снега.

Основные шаги в проекте:

1. Сбор и подготовка данных.
2. Анализ и визуализация данных.
3. Обучение моделей регрессии.
4. Оценка качества моделей.
5. Применение модели для предсказания средней температуры.

Таким образом, конечной целью проекта является создание точной модели предсказания средней температуры.

2. Описание датасета

Датасет представляет собой реальный набор данных, собранный на основе метеорологических измерений, проводимых ежедневно в течение нескольких лет. Эти данные включают разнообразные показатели, характеризующие погодные условия на конкретную дату.

Общая информация:

Датасет состоит из 15341 наблюдения, каждое из которых описывается 10 столбцами. Каждый столбец представляет собой определенное метеорологическое измерение, которое было записано на конкретную дату.

Столбцы датасета:

- **Дата (date)**: дата измерений, представлена в формате ГГГГММДД (например, 19790101 для 1 января 1979 года).
- **Облачный покров (cloud_cover)**: измерение облачного покрова в октах. Окта - это единица измерения количества облаков, где 0 означает ясное небо, а 8 - полностью облачное небо.
- **Солнечный свет (sunshine)**: продолжительность солнечного света в часах.
- **Глобальное излучение (global_radiation)**: измерение освещенности в ваттах на квадратный метр (Вт/м²).
- **Максимальная температура (max_temp)**: максимальная температура в течение дня, измеренная в градусах Цельсия.
- **Средняя температура (mean_temp)**: средняя температура в течение дня, измеренная в градусах Цельсия.
- **Минимальная температура (min_temp)**: минимальная температура в течение дня, измеренная в градусах Цельсия.
- **Атмосферные осадки (precipitation)**: количество осадков в течение дня, измеренное в миллиметрах.
- **Давление (pressure)**: атмосферное давление, измеренное в паскалях.
- **Глубина снега (snow_depth)**: глубина снежного покрова в сантиметрах.

Пример нескольких строк из датасета для наглядности:

```
+--------+-----------+--------+----------------+--------+---------+--------+-------------+--------+----------+
|    date|cloud_cover|sunshine|global_radiation|max_temp|mean_temp|min_temp|precipitation|pressure|snow_depth|
+--------+-----------+--------+----------------+--------+---------+--------+-------------+--------+----------+
|19790101|        2.0|     7.0|            52.0|     2.3|     -4.1|    -7.5|          0.4|101900.0|       9.0|
|19790102|        6.0|     1.7|            27.0|     1.6|     -2.6|    -7.5|          0.0|102530.0|       8.0|
|19790103|        5.0|     0.0|            13.0|     1.3|     -2.8|    -7.2|          0.0|102050.0|       4.0|
|19790104|        8.0|     0.0|            13.0|    -0.3|     -2.6|    -6.5|          0.0|100840.0|       2.0|
|19790105|        6.0|     2.0|            29.0|     5.6|     -0.8|    -1.4|          0.0|102250.0|       1.0|
|19790106|        5.0|     3.8|            39.0|     8.3|     -0.5|    -6.6|          0.7|102780.0|       1.0|
|19790107|        8.0|     0.0|            13.0|     8.5|      1.5|    -5.3|          5.2|102520.0|       0.0|
|19790108|        8.0|     0.1|            15.0|     5.8|      6.9|     5.3|          0.8|101870.0|       0.0|
|19790109|        4.0|     5.8|            50.0|     5.2|      3.7|     1.6|          7.2|101170.0|       0.0|
|19790110|        7.0|     1.9|            30.0|     4.9|      3.3|     1.4|          2.1| 98700.0|       0.0|
+--------+-----------+--------+----------------+--------+---------+--------+-------------+--------+----------+
only showing top 10 rows
```

3. Ход работы

3.1. Гипотеза

Исходя из данных о погоде за несколько лет, мы предположили, что средняя температура может быть предсказана с использованием регрессионных моделей, так как многие признаки могут иметь нелинейные зависимости с температурой.

3.2. Выбор модели регрессии

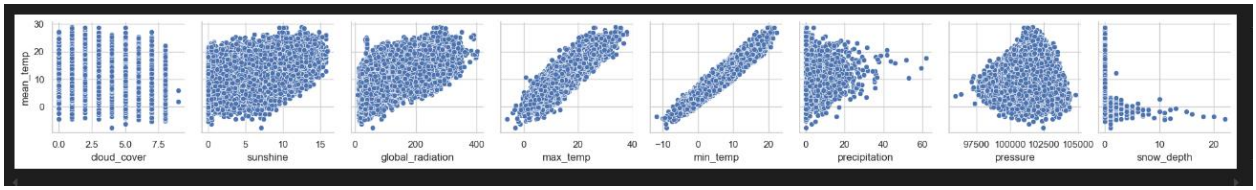
С помощью анализа данных с использованием Python было выявлено, что между признаками и целевой переменной (средняя температура) существуют нелинейные зависимости. Это было установлено путем визуализации данных и анализа графиков

```python
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

df = pd.read_csv('london_weather.csv')
sns.set_style(style="whitegrid")
pairplot = sns.pairplot(df, x_vars=['cloud_cover', 'sunshine', 'global_radiation', 'max_temp', 'min_temp', 'precipitation', 'pressure',

plt.show()
```

✓ 2.7s @screenshoter

```python
ine', 'global_radiation', 'max_temp', 'min_temp', 'precipitation', 'pressure', 'snow_depth'], y_vars='mean_temp', kind='scatter')
```

✓ 2.7s Python



Таким образом, для того чтобы учесть и корректно моделировать такие нелинейные зависимости, было принято решение использовать специализированные методы регрессии. Эти методы могут эффективно работать с нелинейными данными, улавливая сложные взаимосвязи между признаками и целевой переменной.

Решено использовать следующие регрессионные модели, способные работать с нелинейными зависимостями:

1. Decision Tree Regressor (Дерево решений для регрессии): Эта модель хорошо справляется с задачами, где данные имеют сложные нелинейные зависимости.
2. Random Forest Regressor (Случайный лес для регрессии): Этот метод объединяет множество деревьев решений для повышения точности предсказаний и уменьшения переобучения.
3. Gradient-Boosted Trees Regressor (Градиентный бустинг деревьев для регрессии): Этот метод улучшает точность модели путем последовательного обучения новых деревьев на ошибках предыдущих.
4. Polynomial Regression (Полиномиальная регрессия) П улавливать более сложные зависимости между признаками и целевой переменной.

Для оценки качества каждой модели использовались стандартные метрики:

1. R^2 (коэффициент детерминации): измеряет объясненную дисперсию модели.
2. MAE (средняя абсолютная ошибка): измеряет среднее абсолютное отклонение между предсказанными и фактическими значениями.
3. MSE (среднеквадратическая ошибка): измеряет среднеквадратическое отклонение между предсказанными и фактическими значениями.

3.3. Код

In [13]:
```
import pyspark
from pyspark.sql.types import StructType, StructField, StringType, FloatType

schema = StructType([
StructField("date", StringType(), True),
StructField("cloud_cover", FloatType(), True),
StructField("sunshine", FloatType(), True),
StructField("global_radiation", FloatType(), True),
StructField("max_temp", FloatType(), True),
StructField("mean_temp", FloatType(), True),
StructField("min_temp", FloatType(), True),
StructField("precipitation", FloatType(), True),
StructField("pressure", FloatType(), True),
StructField("snow_depth", FloatType(), True)
])

df = spark.read.csv("file:///home/student/london_weather.csv", header=True, schema=schema)
df.printSchema()
df.show(10)
```

```
root
 |-- date: string (nullable = true)
 |-- cloud_cover: float (nullable = true)
 |-- sunshine: float (nullable = true)
 |-- global_radiation: float (nullable = true)
 |-- max_temp: float (nullable = true)
 |-- mean_temp: float (nullable = true)
 |-- min_temp: float (nullable = true)
 |-- precipitation: float (nullable = true)
 |-- pressure: float (nullable = true)
 |-- snow_depth: float (nullable = true)
```

```
In [22]: df=df.dropna()

In [24]: names = []
         mses = []
         maes = []
         r2s = []

In [25]: from pyspark.ml.feature import VectorAssembler
         from pyspark.ml.evaluation import RegressionEvaluator
         from pyspark.ml.regression import DecisionTreeRegressor

         feature_columns = [col for col in df.columns if col not in ['date', 'mean_temp']]
         train_df, test_df = df.randomSplit([0.7, 0.3])

         assembler = VectorAssembler(inputCols=feature_columns, outputCol='features')
         train_df_transform = assembler.transform(train_df).select('features', 'mean_temp')
         test_df_transform = assembler.transform(test_df).select('features', 'mean_temp')

         evaluator_mse = RegressionEvaluator(labelCol='mean_temp', predictionCol='prediction', metricName='mse')
         evaluator_mae = RegressionEvaluator(labelCol='mean_temp', predictionCol='prediction', metricName='mae')
         evaluator_r2 = RegressionEvaluator(labelCol='mean_temp', predictionCol='prediction', metricName='r2')

         # Decision Tree Regressor
         dt = DecisionTreeRegressor(featuresCol='features', labelCol='mean_temp')
         dt_model = dt.fit(train_df_transform )
         dt_predictions = dt_model.transform(test_df_transform)
         dt_mse = evaluator_mse.evaluate(dt_predictions)
         dt_mae = evaluator_mae.evaluate(dt_predictions)
         dt_r2 = evaluator_r2.evaluate(dt_predictions)
         print(f"Decision Tree Regressor MSE = {round(dt_mse, 10)}, MAE = {round(dt_mae, 10)}, R^2 = {round(dt_r2, 10)}")

         names.append("Decision Tree Regressor")
         mses.append(dt_mse)
         maes.append(dt_mae)
         r2s.append(dt_r2)

         Decision Tree Regressor MSE = 1.6408667843, MAE = 1.0104671563, R^2 = 0.9498521328
```

Данный код проводит полный цикл обучения модели регрессии на основе решающего дерева с использованием PySpark, начиная от подготовки данных и преобразования признаков до оценки и вывода результатов.

Полученные значения MSE, MAE и R^2 выводятся в консоль, а также сохраняются для последующего сравнения с другими моделями.

```
In [30]: from pyspark.ml.regression import GBTRegressor
         # Gradient-Boosted Trees Regressor
         gbt = GBTRegressor(featuresCol='features', labelCol='mean_temp')
         gbt_model = gbt.fit(train_df_transform)
         gbt_predictions = gbt_model.transform(test_df_transform)
         gbt_mse = evaluator_mse.evaluate(gbt_predictions)
         gbt_mae = evaluator_mae.evaluate(gbt_predictions)
         gbt_r2 = evaluator_r2.evaluate(gbt_predictions)
         print(f"Gradient-Boosted Trees Regressor MSE = {round(gbt_mse, 10)}, MAE = {round(gbt_mae, 10)}, R^2 = {round(gbt_r

         names.append("Gradient-Boosted Trees Regressor")
         mses.append(gbt_mse)
         maes.append(gbt_mae)
         r2s.append(gbt_r2)

         2024-06-17 05:12:38,017 WARN netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSyste
         mBLAS
         2024-06-17 05:12:38,026 WARN netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBL
         AS
         [Stage 269:>                                                        (0 + 1) / 1]

         Gradient-Boosted Trees Regressor MSE = 1.1320844383, MAE = 0.8376575933, R^2 = 0.9654014448
```


Этот код использует библиотеку PySpark для создания и обучения модели, известной как случайный лес (Random Forest), с целью предсказания средней температуры (mean_temp).
Результаты оценки выводятся для анализа производительности модели.

In [27]:
```
from pyspark.ml.regression import RandomForestRegressor

# Random Forest Regressor
rf = RandomForestRegressor(featuresCol='features', labelCol='mean_temp')
rf_model = rf.fit(train_df_transform)
rf_predictions = rf_model.transform(test_df_transform)
rf_mse = evaluator_mse.evaluate(rf_predictions)
rf_mae = evaluator_mae.evaluate(rf_predictions)
rf_r2 = evaluator_r2.evaluate(rf_predictions)
print(f"Random Forest Regressor MSE = {round(rf_mse, 10)}, MAE = {round(rf_mae, 10)}, R^2 = {round(rf_r2, 10)}")

names.append("Random Forest Regressor")
mses.append(rf_mse)
maes.append(rf_mae)
r2s.append(rf_r2)
```

```
Random Forest Regressor MSE = 1.4805811765, MAE = 0.9164286229, R^2 = 0.9547507519
```


Данный код включает в себя полный цикл работы с моделью градиентного бустинга в PySpark, начиная от её создания и обучения до оценки её производительности на тестовых данных.

Результаты оценки модели выводятся на экран для дальнейшего анализа и сравнения с другими моделями.

In [33]:
```
from pyspark.ml.feature import PolynomialExpansion
from pyspark.ml.regression import LinearRegression

df_p = assembler.transform(df)
train_df_p, test_df_p = df_p.randomSplit([0.7, 0.3])

# Polynomial Regressor
poly = PolynomialExpansion(degree=2, inputCol='features', outputCol='poly_features')
train_poly_transform = poly.transform(train_df_p).select('poly_features', 'mean_temp')
test_poly_transform = poly.transform(test_df_p).select('poly_features', 'mean_temp')
lr = LinearRegression(featuresCol='poly_features', labelCol='mean_temp')
lr_model = lr.fit(train_poly_transform)
lr_predictions = lr_model.transform(test_poly_transform)
poly_mse = evaluator_mse.evaluate(lr_predictions)
poly_mae = evaluator_mae.evaluate(lr_predictions)
poly_r2 = evaluator_r2.evaluate(lr_predictions)
print(f"Polynomial Regressor MSE = {round(poly_mse, 10)}, MAE = {round(poly_mae, 10)}, R^2 = {round(poly_r2, 10)}")

names.append("Polynomial Regressor")
mses.append(poly_mse)
maes.append(poly_mae)
r2s.append(poly_r2)
```

```
2024-06-17 05:52:25,131 WARN util.Instrumentation: [c6951ff8] regParam is zero, which might cause numerical instab
ility and overfitting.
2024-06-17 05:53:29,251 WARN netlib.LAPACK: Failed to load implementation from: com.github.fommil.netlib.NativeSys
temLAPACK
2024-06-17 05:53:29,336 WARN netlib.LAPACK: Failed to load implementation from: com.github.fommil.netlib.NativeRef
LAPACK
```

```
Polynomial Regressor MSE = 0.7324238477, MAE = 0.6657779766, R^2 = 0.9774369963
```

Этот код использует PySpark для создания и обучения модели полиномиальной регрессии второй степени на данных о средней температуре

Данные подготавливаются с использованием VectorAssembler, а затем разделяются на обучающий и тестовый наборы. После этого применяется PolynomialExpansion для создания полиномиальных признаков второй степени. Модель линейной регрессии настраивается на этих данных и обучается на обучающем наборе. Затем модель используется для предсказания на тестовом наборе данных, после чего оцениваются её производительность с помощью метрик MSE, MAE и R^2, результаты которых выводятся на экран.

```
In [34]: print(names)
         print(mses)
         print(maes)
         print(r2s)

['Decision Tree Regressor', 'Random Forest Regressor', 'Gradient-Boosted Trees Regressor', 'Polynomial Regressor']
[1.6408667842766052, 1.4805811765486476, 1.132084438308594, 0.7324238477310435]
[1.0104671562885021, 0.9164286228687513, 0.8376575933055127, 0.665777976586681]
[0.94985213284269, 0.9547507519387636, 0.9654014447929813, 0.9774369963398843]

In [ ]:
```

Этот код выводит четыре списка: names, mses, maes и r2s, содержащие информацию о моделях и их соответствующих метриках производительности.

Список names: выводит названия четырех моделей регрессии: Decision Tree Regressor, Random Forest Regressor, Gradient-Boosted Trees Regressor и Polynomial Regressor.

Список mses: содержит значения среднеквадратичной ошибки (MSE) для каждой модели. В данном случае: 1.6408667842766052, 1.4805811765486476, 1.132084438308594 и 0.7324238477310435 соответственно.

Список maes: выводит значения средней абсолютной ошибки (MAE) для каждой модели: 1.0104671562885021, 0.9164286228687513, 0.8376575933055127 и 0.6657779765866811 соответственно.

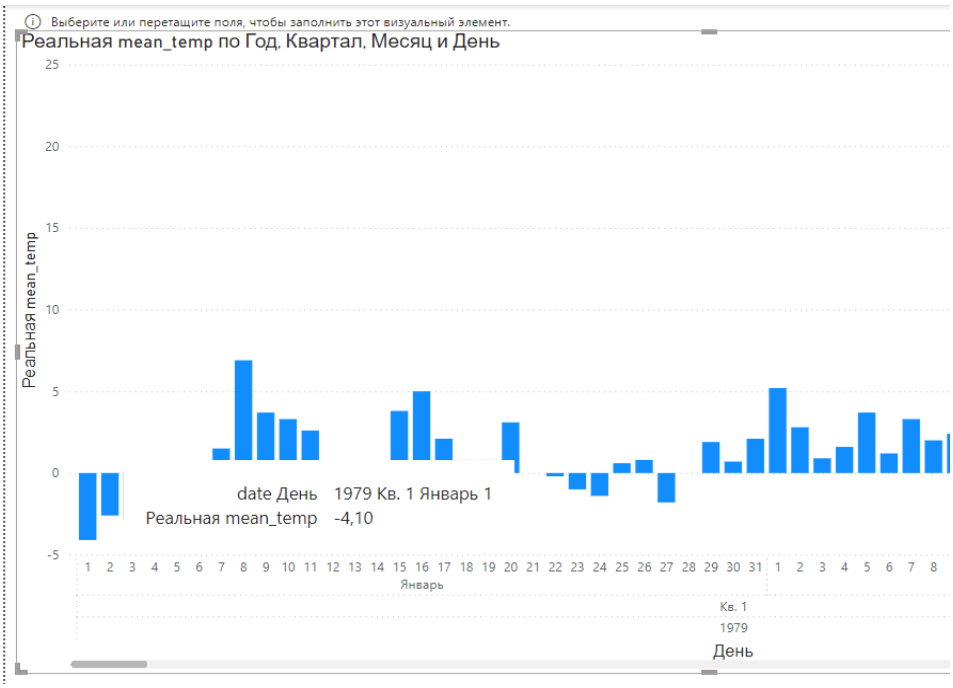
Список r2s: содержит значения коэффициента детерминации (R^2) для каждой модели: 0.94985213284269, 0.9547507519387636, 0.9654014447929813 и 0.9774369963398843 соответственно.

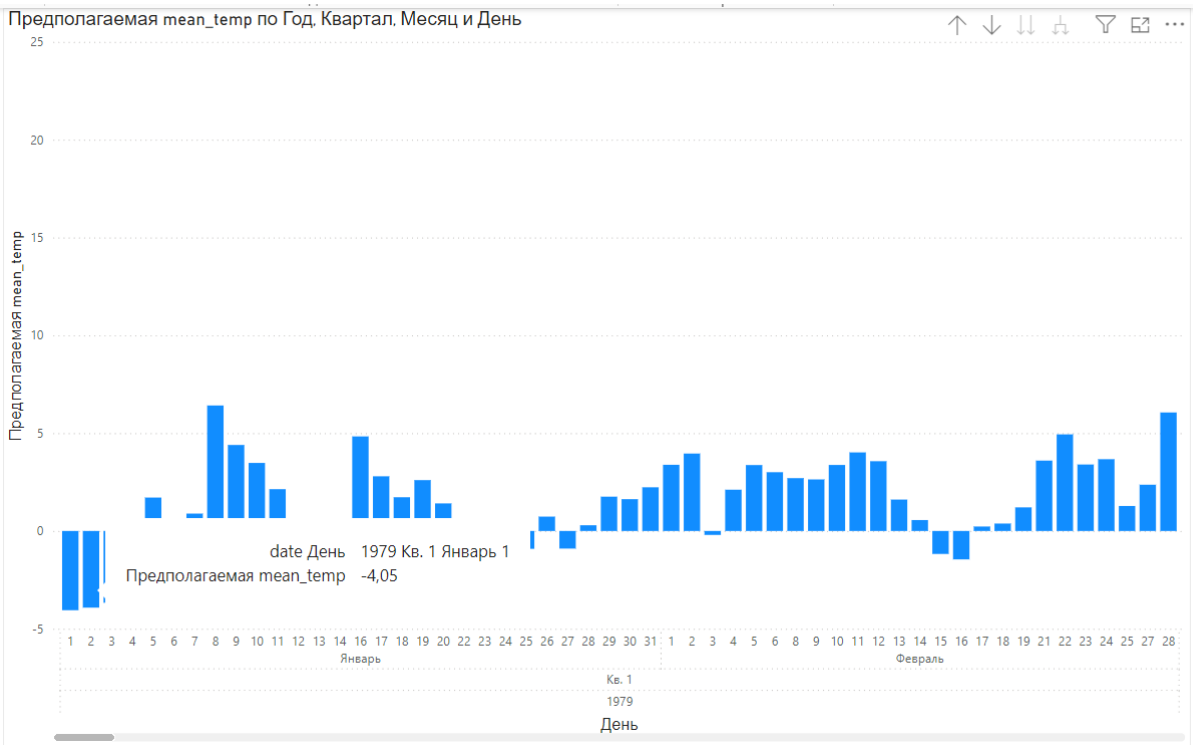
```
In [38]: lr_predictions.show(20)

+--------------------+---------+--------------------+
|       poly_features|mean_temp|          prediction|
+--------------------+---------+--------------------+
|[6.0,36.0,1.70000...|     -2.6| -3.7678904345240483|
|[8.0,64.0,0.0,0.0...|     -2.6|  -3.466720970823303|
|[5.0,25.0,3.79999...|     -0.5| -0.1951177839715399|
|[8.0,64.0,0.0,0.0...|      1.5|  0.8964442189587487|
|[3.0,9.0,6.400000...|      0.4| 0.19700736033664157|
|[7.0,49.0,0.0,0.0...|     -0.6| -0.4828913486861097|
|[8.0,64.0,0.0,0.0...|      0.8|  2.6178029277631083|
|[7.0,49.0,0.0,0.0...|     -1.0|-0.24939434181271736|
|[4.0,16.0,1.0,4.0...|      1.9|  1.8248672806650745|
|[5.0,25.0,1.60000...|      2.1|   2.198888954718228|
|[5.0,25.0,1.39999...|      2.8|   3.956527946990917|
|[1.0,1.0,8.699999...|      0.9|-0.34414403219258816|
|[8.0,64.0,0.10000...|      1.6|   2.035654997305727|
|[6.0,36.0,1.89999...|      2.0|   2.670250781759677|
|[8.0,64.0,0.0,0.0...|      2.8|  3.2915613855538766|
|[6.0,36.0,2.40000...|     -0.2|  0.4885466911167953|
|[8.0,64.0,0.0,0.0...|     -0.6|    0.36180305734797|
|[6.0,36.0,2.09999...|      4.4|   6.044792843639101|
|[4.0,16.0,8.69999...|      7.4|   8.246131283285848|
|[4.0,16.0,6.40000...|      5.4|   4.444119435923739|
+--------------------+---------+--------------------+
only showing top 20 rows
```

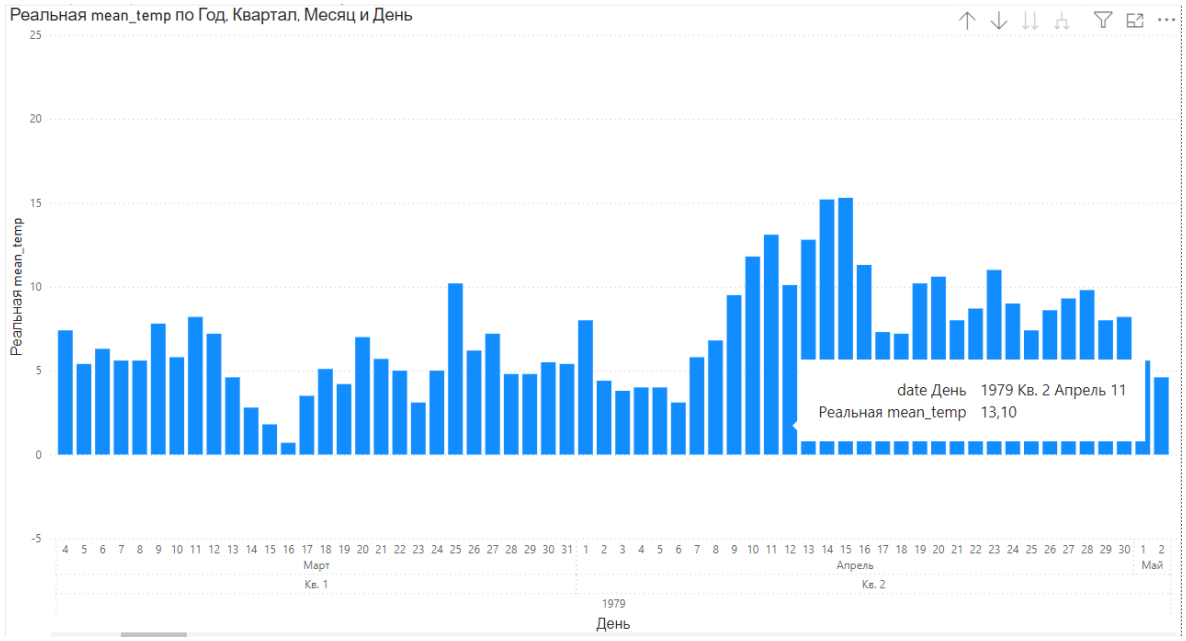
В выводе `lr_predictions.show(20)` представлены результаты предсказаний модели линейной регрессии

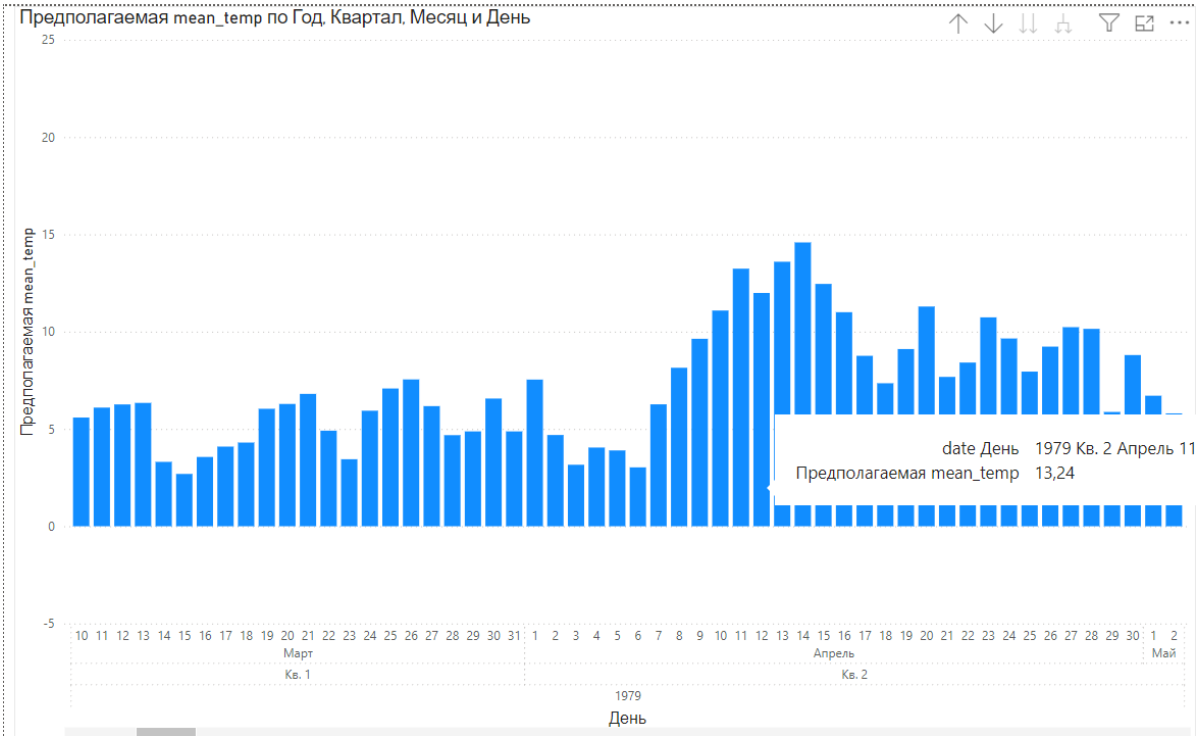
3.4. Визуализация данных

Для наглядного представления данных были созданы гистограммы в Power BI, отображающие изменения средней температуры по датам.


Выберите или перетащите поля, чтобы заполнить этот визуальный элемент.
Реальная mean_temp по Год, Квартал, Месяц и День
Реальная mean_temp
25
20
15
10
5
0
-5
date День 1979 Кв. 1 Январь 1
Реальная mean_temp -4,10
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8
Январь
Кв. 1
1979
День


Предполагаемая mean_temp по Год, Квартал, Месяц и День
Предполагаемая mean_temp
25
20
15
10
5
0
-5
date День 1979 Кв. 1 Январь 1
Предполагаемая mean_temp -4,05
1 2 3 4 5 6 7 8 9 10 11 12 13 14 16 17 18 19 20 22 23 24 25 26 27 28 29 30 31 1 2 3 4 5 6 8 9 10 11 12 13 14 15 16 17 18 19 21 22 23 24 25 27 28
Январь
Февраль
Кв. 1
1979
День





Сравнение двух наборов данных выявило незначительные различия, однако они находятся в пределах допустимых значений. В целом, результаты нашего анализа подтверждают основную тенденцию, наблюдаемую в исходных данных, и подтверждают достоверность проведенного анализа.

4. Выводы

В ходе решения данного кейса, было сделано:

- Проанализирован датасет, отобраны наиболее информативные данные
- Выбраны подходящие регрессионные модели для работы с нелинейными зависимостями.

- Проведено обучение и тестирование выбранных моделей для достижения наилучших результатов в прогнозировании средней температуры.
- Построены гистограммы в Power BI для сравнения результатов.