ОТЧЕТ

Здание№2  
Вариант 4

«Обучение нейросетевых моделей анализа последовательностей»

Студентка группы №932001: Ли Алина

**Цель работы** – получить навыки создания нейронных сетей для анализа сигналов с помощью моделей рекуррентных сетей и ячеек LSTM (GRU).

**Задачи** – Построить три нейросетевые модели:

**a) Регрессор, использующий слой RNN в качестве скрытого слоя**

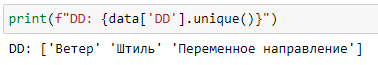
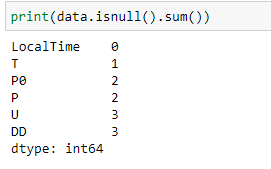
**b) Регрессор, ячейку LSTM (или GRU) в скрытом слое**

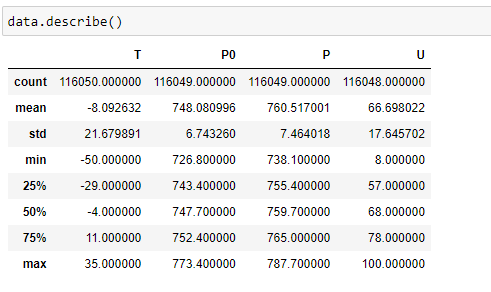
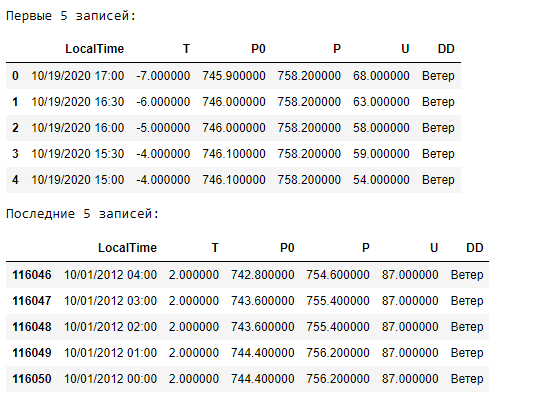
**c) Регрессор – стек двух слоёв с LSTM (или GRU)**

для прогноза средней температуры в Якутске на следующий день, используя информацию за предыдущие дни.

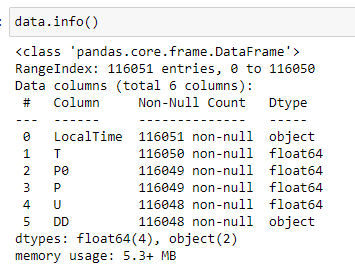
**Описание данных**

1. **LocalTime:** Время наблюдения в формате день/месяц/год час: минута. Это объектный тип данных. В этом наборе данных отсутствуют пропущенные значения.
2. **T:** Температура. Это числовой тип данных (float64). Средняя температура в этом наборе данных составляет -8.09 градусов Цельсия, но она варьируется от -50 до 35 градусов. В этом столбце есть 1 пропущенное значение.
3. **P0:** Давление на уровне моря. Это числовой тип данных (float64). Среднее значение составляет 748.08, но значения варьируются от 726.8 до 773.4. В этом столбце есть 2 пропущенных значения.
4. **P:** Давление в мм ртутного столба. Это числовой тип данных (float64). Среднее значение составляет 760.52, но значения варьируются от 738.1 до 787.7. В этом столбце есть 2 пропущенных значения.
5. **U:** Относительная влажность воздуха. Это числовой тип данных (float64). Среднее значение составляет 66.7%, но значения варьируются от 8% до 100%. В этом столбце есть 3 пропущенных значения.
6. **DD:** Направление ветра. Может принимать значения "Ветер", "Штиль" или "Переменное направление". Это объектный тип данных. В этом столбце есть 3 пропущенных значения.



В некоторых столбцах есть пропущенные значения. Всего в наборе данных 116051 записей.

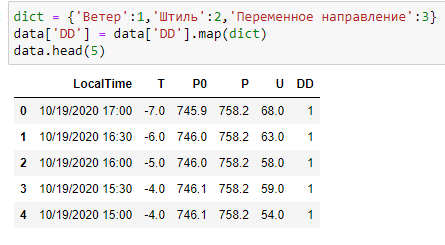


**Подготовка данных**

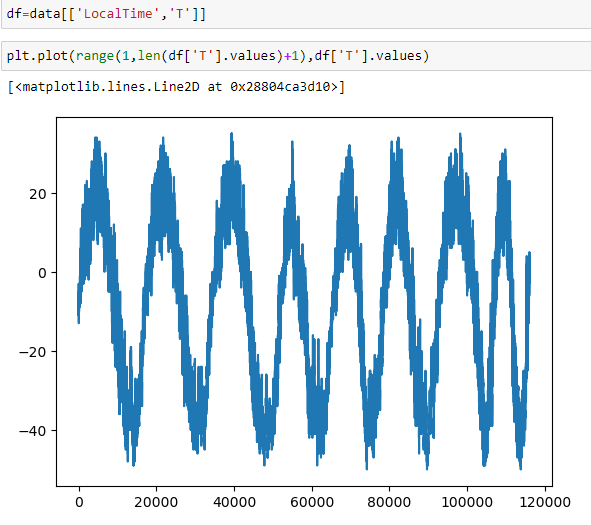
Для обработки пропущенных значений используем комманду

data = data.dropna() (удаляет все строки с пропущенными значениями из набора данных).

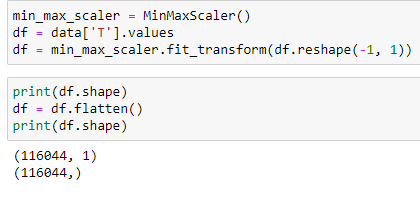
Далее преобразуем категориальные значения в столбце DD в числовые. Создаем словарь, где каждому уникальному значению в столбце DD сопоставлено уникальное число. Затем вы используем метод map(), чтобы заменить все значения в столбце DD на соответствующие числовые значения из словаря.



Создаем новый DataFrame df и строим график, который будет отображать изменение температуры в городе в зависимости от времени. Таким образом, каждая точка на графике будет соответствовать температуре в определенный час определенного дня. Ось Y направлена вверх, и более высокие значения (более теплая температура) расположены выше, а более низкие значения (холодная температура) расположены ниже.



Чтобы нейросеть обрабатывала значения, нам нужно их отмасштабировать Применение MinMaxScaler используется для нормализации данных, что означает приведение значений к определенному диапазону от 0 до 1. Далее данные извлекаются из столбца 'T', преобразуются и затем выравниваются в одномерный массив.

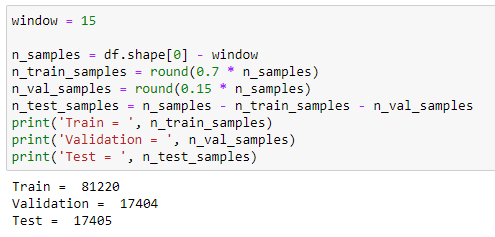


Далее определяем размер окна как 15, что означает, что модель будет использовать данные за последние 15 дней для прогнозирования температуры на следующий день.  
n\_samples = data.shape[0] - window: Вычисляет общее количество образцов, которые можно создать, вычитая размер окна из общего числа элементов в данных. Это связано с тем, что для создания каждой последовательной выборки используется окно фиксированного размера.

n\_train\_samples = round(0.7 \* n\_samples): Вычисляет количество образцов для обучающего набора данных, используя 70% от общего числа образцов. Здесь round используется для округления до целого числа, так как количество образцов должно быть целым.

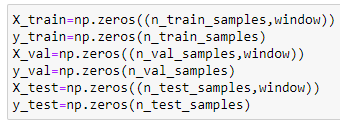
n\_val\_samples = round(0.15 \* n\_samples): Вычисляет количество образцов для валидационного набора данных, используя 15% от общего числа образцов.

n\_test\_samples = n\_samples - n\_train\_samples - n\_val\_samples: Вычисляет количество образцов для тестового набора данных, вычитая количество образцов в обучающем и валидационном наборах из общего числа образцов.

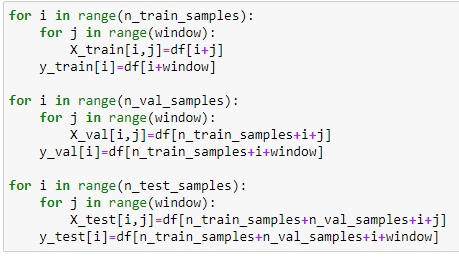


Далее создаем нулевые массивы для хранения данных обучающей, валидационной и тестовой выборок.  
X\_train представляет собой массив размерности (количество образцов в обучающей выборке, размер окна данных ). Аналогично X\_val и X\_test.

y\_train представляет собой массив размерности (n\_train\_samples), где каждый элемент содержит целевое значение (температура) для соответствующего образца в обучающей выборке. Аналогично y\_val и y\_test.   
Изначально массивы содержат только нули.



Далее следующий метод создает последовательные окна данных, где X содержит предыдущие значения (по размеру окна), а y содержит соответствующие следующие значения (целевую переменную).



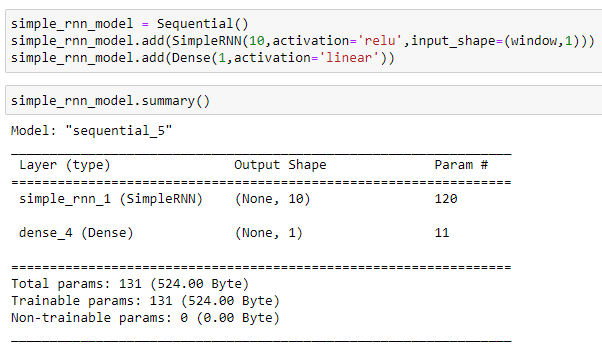
**Регрессор, использующий слой RNN в качестве скрытого слоя**Цель - для прогнозирования средней температуры в городе на следующий день, используя информацию о предыдущих днях. В регрессоре используются слои RNN в качестве скрытого слоя. Эти слои способны учитывать последовательные зависимости в данных.

Для каждого обучающего образца входными данными служат предыдущие значения температуры за окно данных. В первом примере окно = 15.

**Модель№1**

*Слой SimpleRNN:* содержит 10 нейронов и использует функцию активации ReLU. Параметр input\_shape=(window, 1) указывает на форму входных данных. (window - размер окна данных, а 1 - количество признаков.)  
*Dense:* полносвязный слой с одним нейроном, который использует линейную функцию активации. Этот слой генерирует выход модели - прогнозируем одно значение - температуру на следующий день.

Входной формат для этой модели - это трехмерный тензор, состоящий из образцов, окна и признаков. Каждый образец содержит последовательность из 15 предыдущих дней температур (окно), и каждый временной шаг содержит один признак (температура).



Компилируем модель:   
loss=mse: среднеквадратичная ошибка (Mean Squared Error) в качестве функции потерь для обучения модели.

optimizer=adam: оптимизатор Adam для обновления весов модели во время обучения.

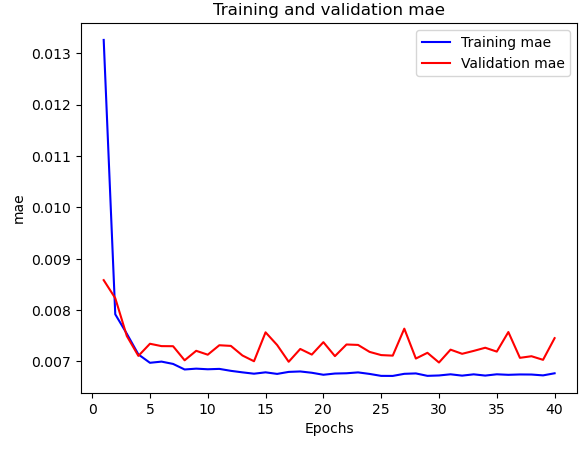
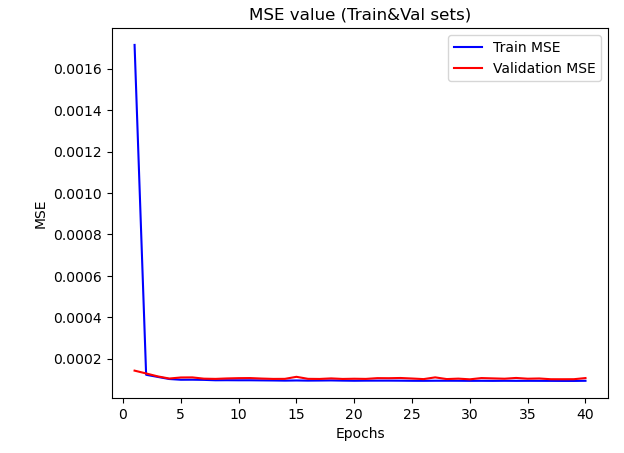
metrics=mae: средняя абсолютная ошибка (Mean Absolute Error) в качестве метрики для оценки производительности модели во время обучения.

**Далее обучаем модель:**

  
batch\_size=25: количество образцов, которые модель обрабатывает за одну итерацию обучения.

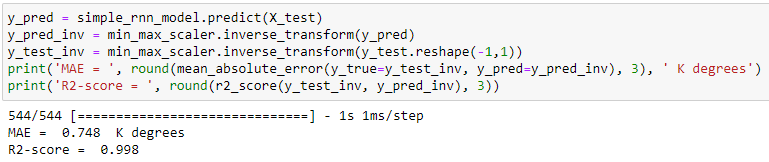
epochs=40: количество полных проходов по обучающим данным.

validation\_data=(X\_val,y\_val): валидационные данные и соответствующие им целевые значения.



Далее предсказываем значения на тестовых данных с использованием обученной модели. Преобразуем предсказанные значения из масштабированного диапазона в исходный. Преобразуем фактические значения тестовых данных обратно из масштабированного диапазона в исходный. Рассчитываем среднюю абсолютную ошибку между фактическими и предсказанными значениями, и коэффициент детерминации между фактическими и предсказанными значениями.  
MAE: 0.748 K degrees. Низкое значение MAE указывает на хорошее качество модели.

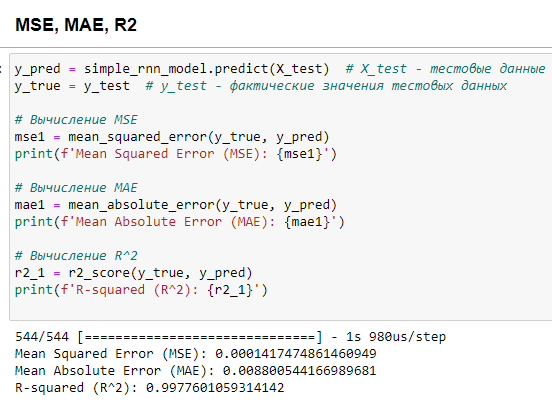
R^2: 0.998. Величина близка к 1, что является отличным показателем.

****

MSE: 0.0001417474861460949, модель в среднем делает очень точные прогнозы.

MAE: 0.008800544166989681, в среднем прогнозы модели отклоняются от истинных значений на очень малую величину.

R^2: 0.9977601059314142

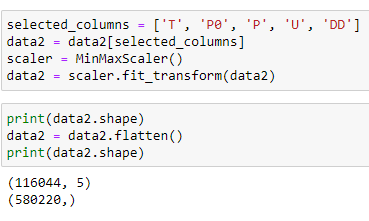
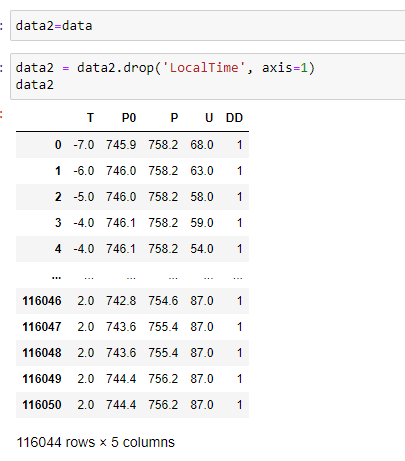
****

**Модель№2**

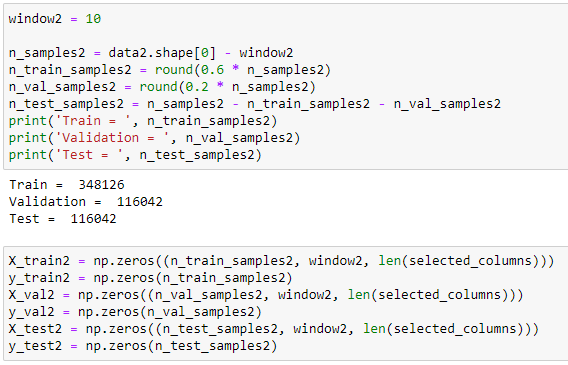
Создадим новый набор данных data2, который в дальнейшем будем использовать для работы со вторыми примерами всех моделей. Для первых примеров используем набор df, который был описан в самом начале.

Из data2 удаляем столбец LocalTime. Затем выбираем определенные столбцы для использования в модели ('T', 'P0', 'P', 'U', 'DD').

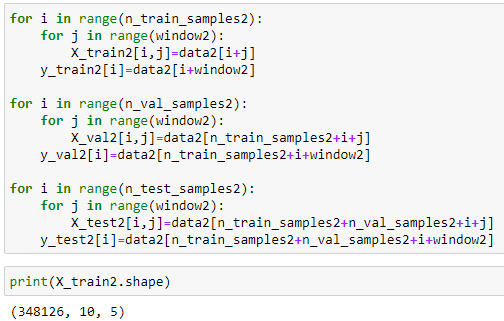
Используем MinMaxScaler для нормализации значений в выбранных столбцах, преобразуем все значения в диапазон от 0 до 1.



Определяем размер окна как 10, что означает, что модель будет использовать данные за последние 10 дней для прогнозирования на следующий день. Затем разделяем данные на обучающий, валидационный и тестовый наборы. 60% от общего числа образцов используется для обучения, 20% - для валидации и оставшиеся 20% - для тестирования.



Далее заполняем ранее инициализированные массивы данными.

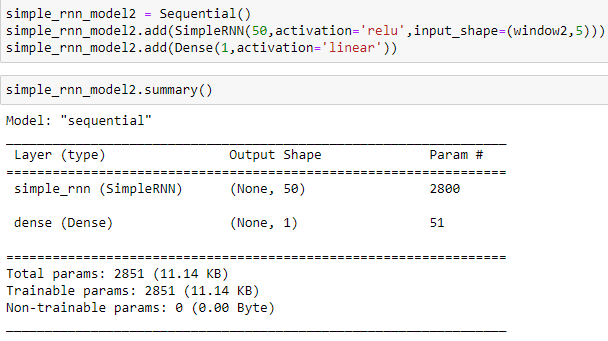


Модель:

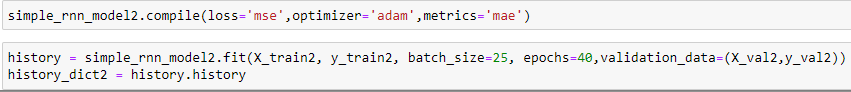
*Слой SimpleRNN:* содержит 50 нейронов и использует функцию активации ReLU.

*Dense:* полносвязный слой с одним нейроном, который использует линейную функцию активации. Этот слой генерирует выход модели - прогнозируем одно значение - температуру на следующий день.

Входной формат для этой модели - это трехмерный тензор, состоящий из образцов, окна и признаков. В данном случае, каждый образец содержит последовательность из 10 предыдущих дней, и каждый временной шаг содержит набор признаков (температура, давление и т.д).

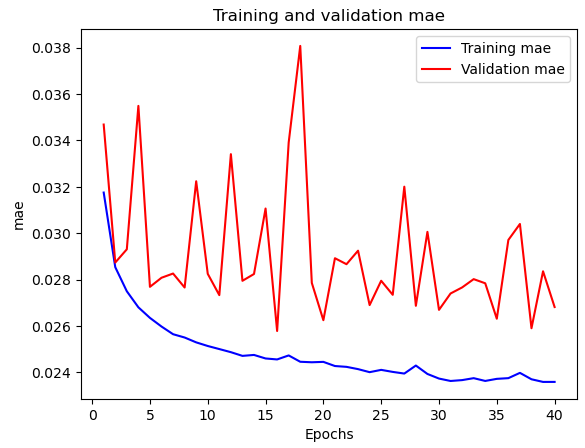
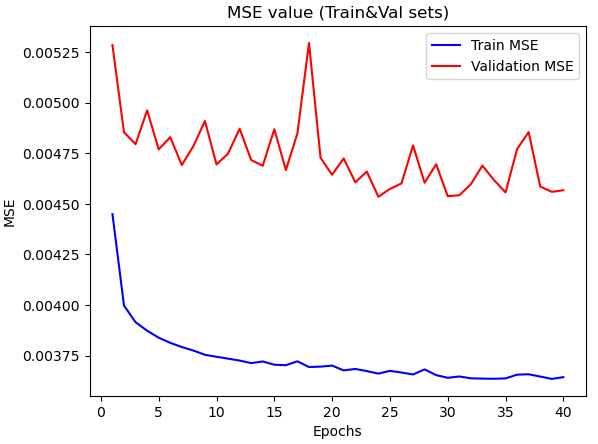


**Далее обучаем модель:**

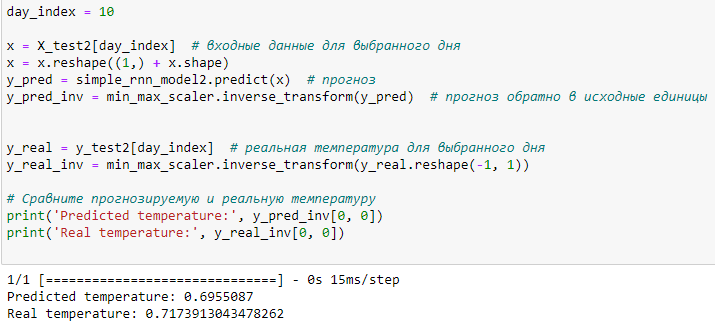
  
batch\_size=25: количество образцов, которые модель обрабатывает за одну итерацию обучения.

epochs=40: количество полных проходов по обучающим данным.

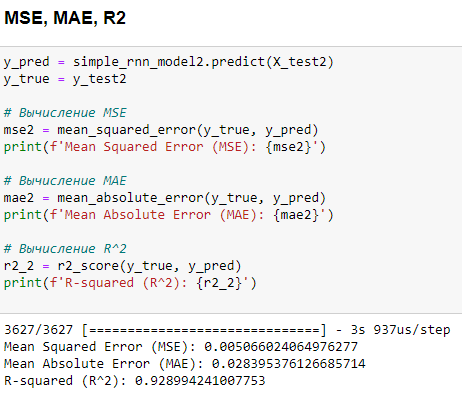
validation\_data=(X\_val2,y\_val2): валидационные данные и соответствующие им целевые значения.



В качестве примера работы обученной модели выберем конкретный день из тестового набора данных и используем вашу обученную модель для прогнозирования температуры на этот день.

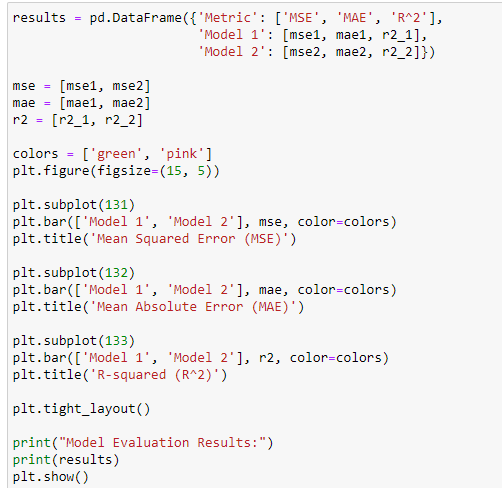


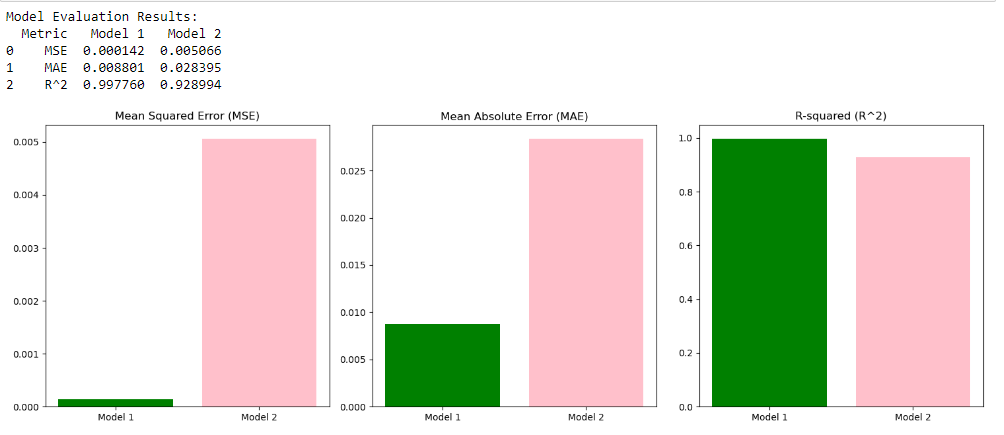
Модель прогнозировала температуру как 0.6955087, в то время как реальная температура составляла 0.7173913043478262. Это означает, что прогноз модели был очень близким к реальной температуре.

****

**Вывод:**

Сравним две модели





Анализ результатов оценки моделей:

**MSE:** Модель 1 имеет MSE равное 0.000142, у модели 2 MSE равно 0.005066. Это означает, что модель 1 в среднем делает более точные прогнозы, чем модель 2.

**MAE:** Модель 1 имеет MAE равное 0.008801, у модели 2 MAE равно 0.028395. Это означает, что в среднем прогнозы модели 1 отклоняются от истинных значений на меньшую величину, чем прогнозы модели 2.

**R^2:** Модель 1 имеет R^2 равное 0.997760, у модели 2 R^2 равно 0.928994. Это означает, что модель 1 лучше объясняет вариацию в данных, чем модель 2.

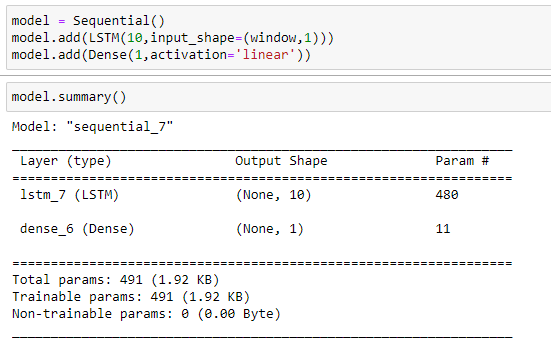
В целом, по всем этим метрикам модель 1 показывает лучшую производительность, чем модель 2.

**Регрессор, ячейку LSTM (или GRU) в скрытом слое**

Цель - для прогнозирования средней температуры в городе на следующий день, используя информацию о предыдущих днях. В этой модели используется ячейка LSTM (или GRU) в скрытом слое, что позволяет учесть зависимости во времени и улавливать долгосрочные зависимости во входных данных.

**Модель№1**

Используем набор данных df.

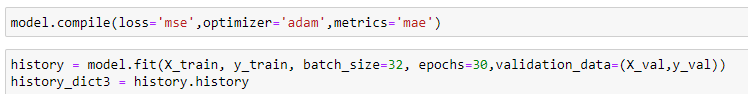


*Слой LSTM:*содержит 10 нейронов, способен учиться долгосрочным зависимостям.

*Dense:* полносвязный слой с одним нейроном, который использует линейную функцию активации. Этот слой генерирует выход модели - прогнозируем одно значение - температуру на следующий день.

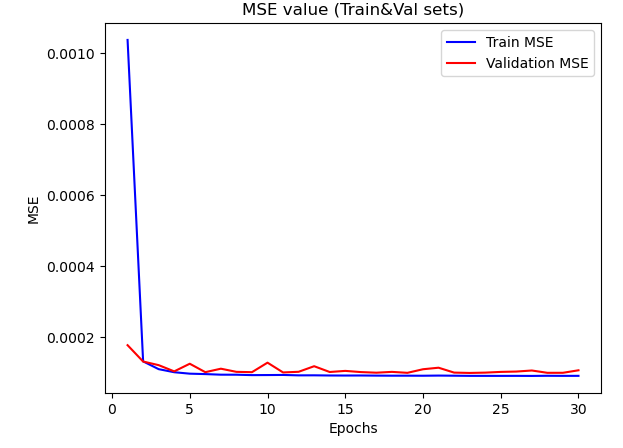
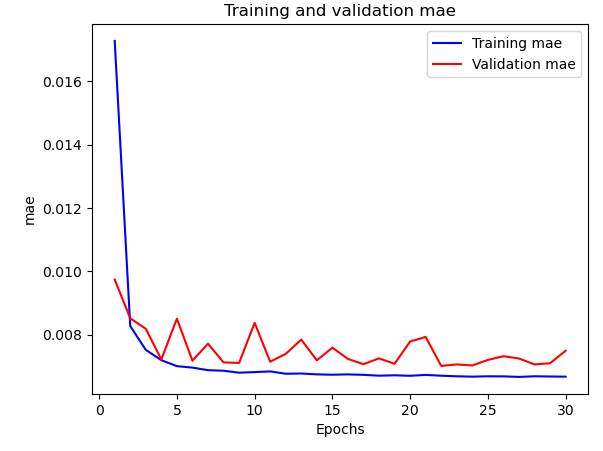
Входной формат для этой модели - это трехмерный тензор, состоящий из образцов, окна и признаков. В данном случае, каждый образец содержит последовательность из 10 предыдущих дней, и каждый временной шаг содержит один признак (температура).

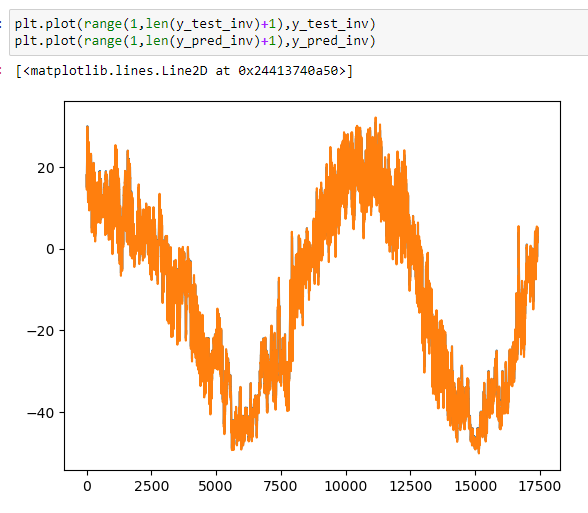
**Далее обучаем модель:**

  
batch\_size=32: количество образцов, которые модель обрабатывает за одну итерацию обучения.

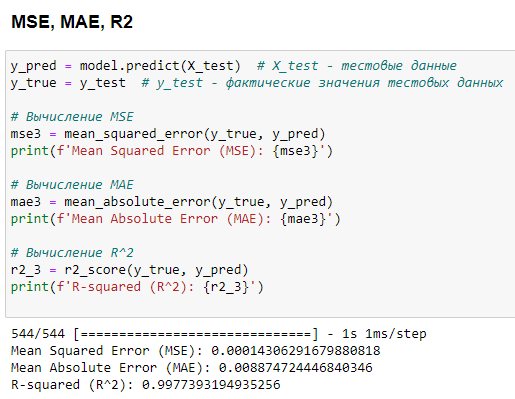
epochs=30: количество полных проходов по обучающим данным.

validation\_data=(X\_val,y\_val): валидационные данные и соответствующие им целевые значения.

В качестве примера работы обученной модели построим слудеющий график:  


Синее - в тестовом наборе, оранжевое - то, что предсказали.



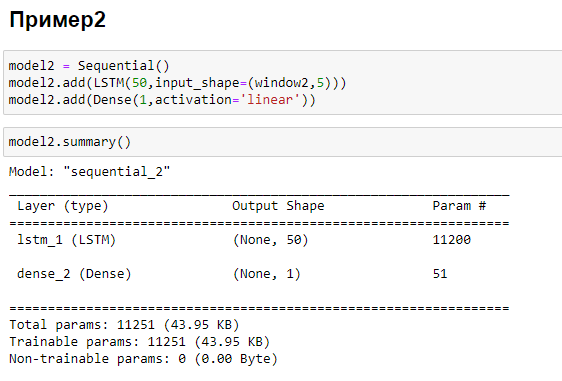
**Модель№2**

Используем набор данных data2.

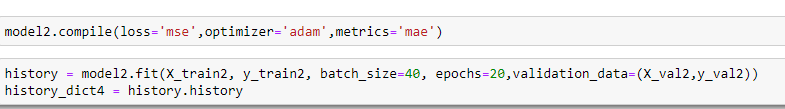
*Слой LSTM:*содержит 50 нейронов, способен учиться долгосрочным зависимостям.

*Dense:* полносвязный слой с одним нейроном, который использует линейную функцию активации. Этот слой генерирует выход модели - прогнозируем одно значение - температуру на следующий день.

Входной формат для этой модели - это трехмерный тензор, состоящий из образцов, окна и признаков. В данном случае, каждый образец содержит последовательность из 10 предыдущих дней, и каждый временной шаг содержит набор признаков (температура, давление и т.д).

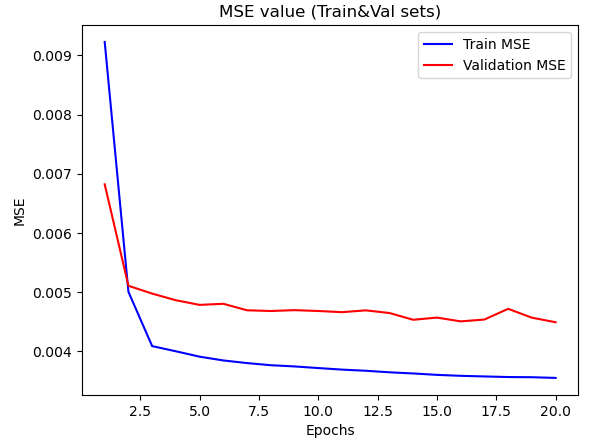
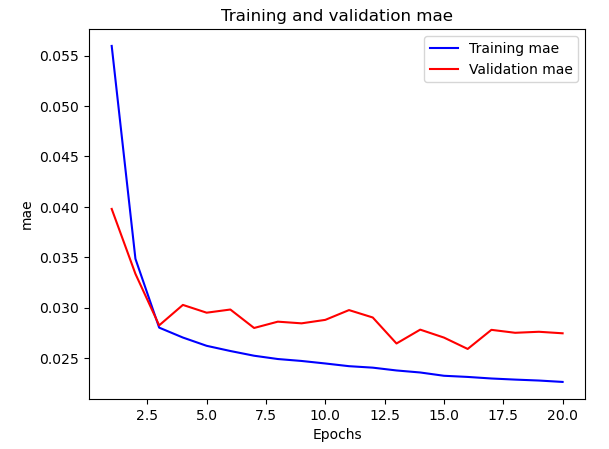


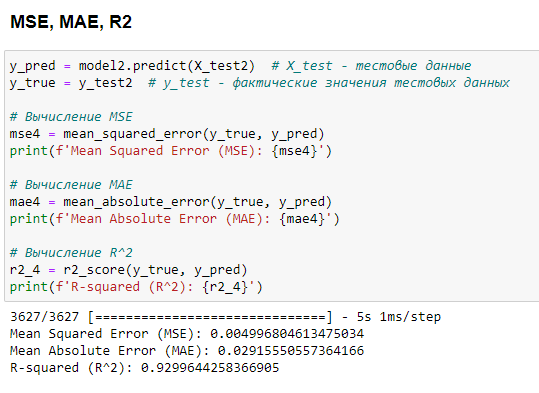
**Далее обучаем модель:**

  
batch\_size=40: количество образцов, которые модель обрабатывает за одну итерацию обучения.

epochs=20: количество полных проходов по обучающим данным.

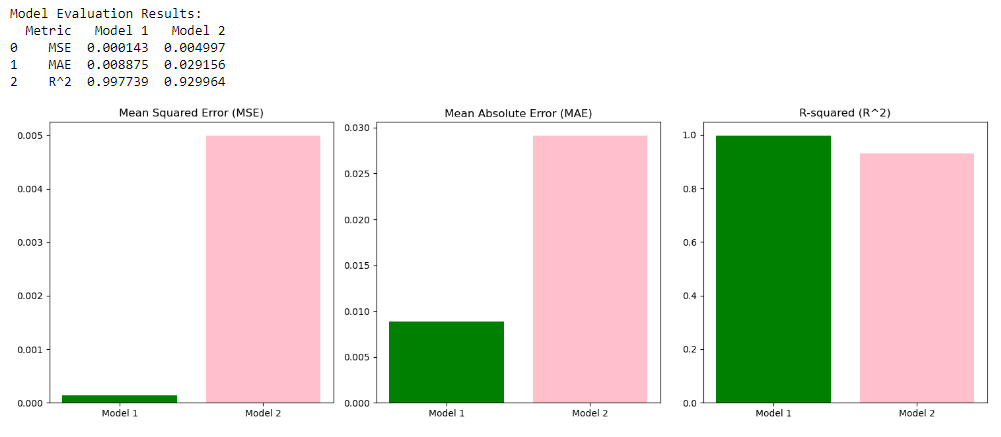
validation\_data=(X\_val2,y\_val2): валидационные данные и соответствующие им целевые значения.



**Вывод:**

Сравним две модели



Анализ результатов оценки моделей:

**MSE:** Модель 1 имеет MSE равное 0.000143, у модели 2 MSE равно 0.004997. Это означает, что модель 1 в среднем делает более точные прогнозы, чем модель 2.

**MAE:** Модель 1 имеет MAE равное 0.008875, у модели 2 MAE равно 0.029156. Это означает, что в среднем прогнозы модели 1 отклоняются от истинных значений на меньшую величину, чем прогнозы модели 2.

**R^2:** Модель 1 имеет R^2 равное 0.997739, у модели 2 R^2 равно 0.929964. Это означает, что модель 1 лучше объясняет вариацию в данных, чем модель 2.

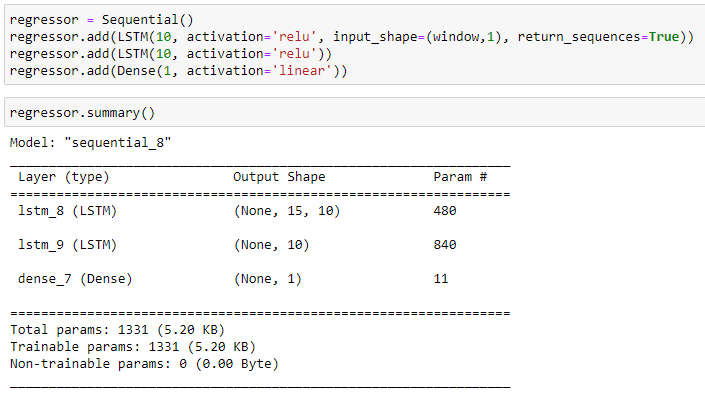
В целом, по всем этим метрикам модель 1 показывает лучшую производительность, чем модель 2.

**Регрессор – стек двух слоёв с LSTM (или GRU)**

Цель - для прогнозирования средней температуры в городе на следующий день, используя информацию о предыдущих днях. Модель представляет собой стек из двух слоев LSTM (или GRU), которые являются рекуррентными слоями. Эти слои обрабатывают временные ряды данных о температуре за предыдущие дни и выдают прогноз на следующий день.

**Модель№1**

Используем набор данных df.



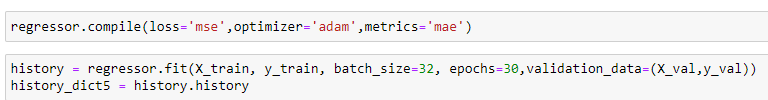
*Слой LSTM:*содержит 10 нейронов, способен учиться долгосрочным зависимостям. Использует функцию активации ReLU.

*Второй LSTM слой:* также содержит 10 нейронов и использует функцию активации ReLU.

*Dense:* полносвязный слой с одним нейроном, который использует линейную функцию активации. Этот слой генерирует выход модели - прогнозируем одно значение - температуру на следующий день.

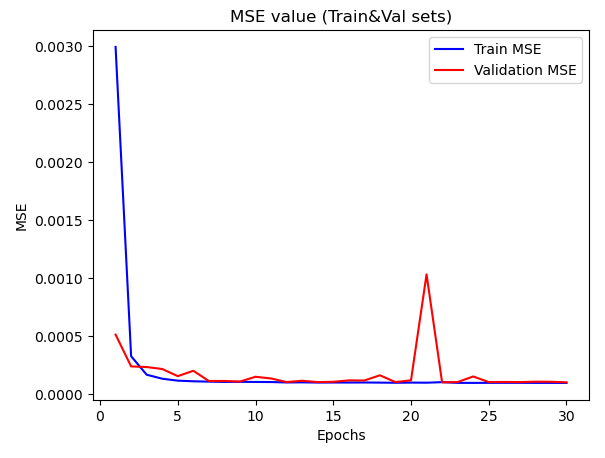
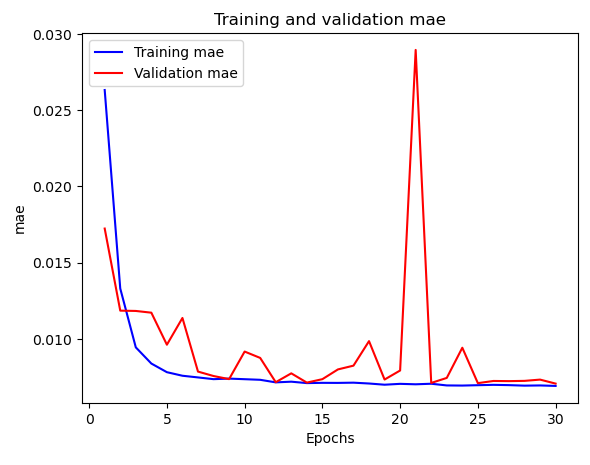
Первый слой обрабатывает данные и передает их второму слою. Второй слой дополнительно анализирует полученную информацию и передает ее на Dense, который выдает окончательный прогноз.

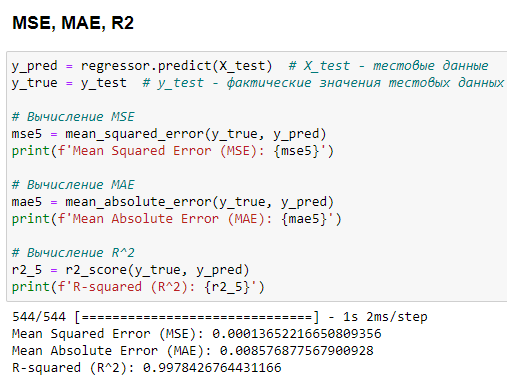
**Далее обучаем модель:**

  
batch\_size=32: количество образцов, которые модель обрабатывает за одну итерацию обучения.

epochs=30: количество полных проходов по обучающим данным.

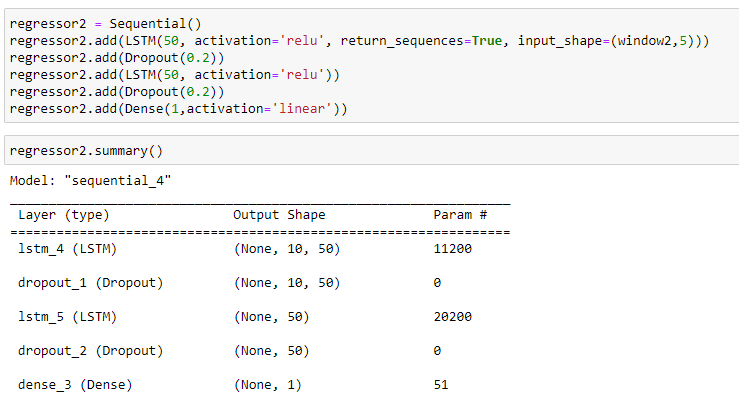
validation\_data=(X\_val,y\_val): валидационные данные и соответствующие им целевые значения.



**Модель№2**

Используем набор данных data2.



*Слой LSTM:*содержит 50 нейронов, способен учиться долгосрочным зависимостям. Использует функцию активации ReLU.

*Слой Dropout:* Этот слой случайным образом “выключает” 20% нейронов в предыдущем слое во время обучения, что помогает предотвратить переобучение.

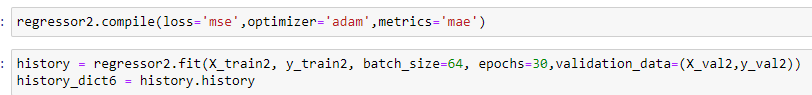
*Второй LSTM слой:* также содержит 10 нейронов и использует функцию активации ReLU.

*Слой Dropout:* Этот слой случайным образом “выключает” 20% нейронов в предыдущем слое во время обучения.

*Dense:* полносвязный слой с одним нейроном, который использует линейную функцию активации. Этот слой генерирует выход модели - прогнозируем одно значение - температуру на следующий день.

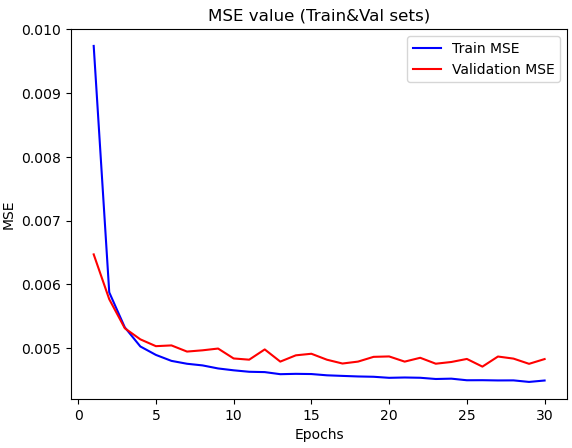
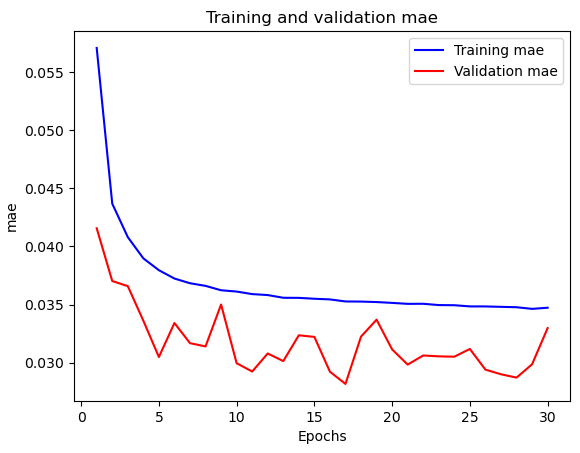
Модель состоит из двух слоев LSTM, которые анализируют временные данные и выявляют закономерности, применяет прореживание для предотвращения переобучения и завершает прогноз средней температуры через плотный слой.

**Далее обучаем модель:**

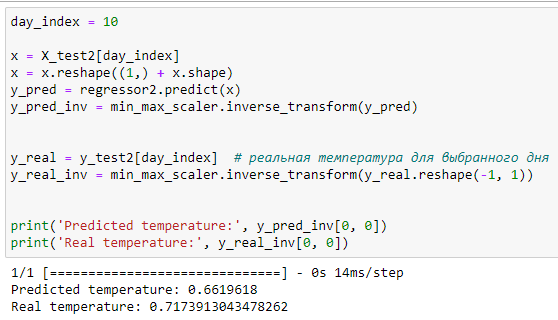
  
batch\_size=64: количество образцов, которые модель обрабатывает за одну итерацию обучения.

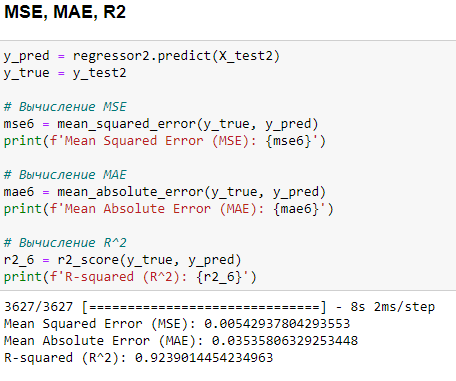
epochs=30: количество полных проходов по обучающим данным.

validation\_data=(X\_val2,y\_val2): валидационные данные и соответствующие им целевые значения.

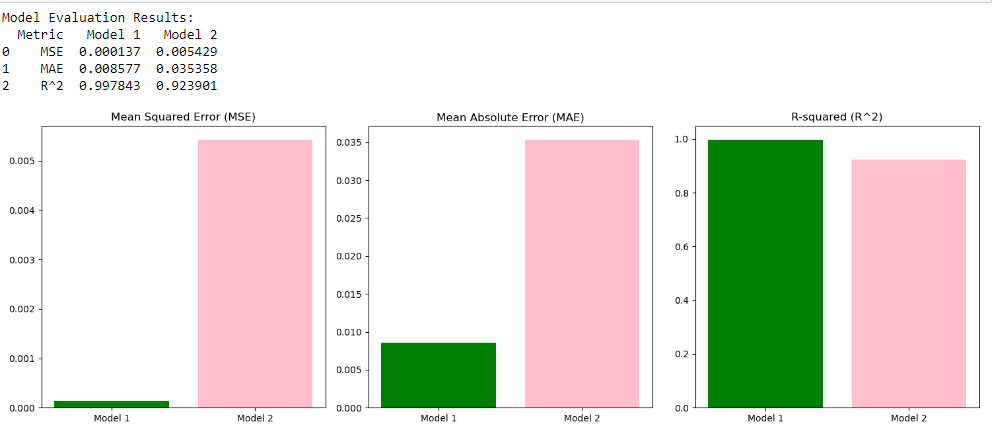
В качестве примера работы обученной модели выберем конкретный день из тестового набора данных и используем вашу обученную модель для прогнозирования температуры на этот день.





**Вывод:**

Сравним две модели



Анализ результатов оценки моделей:

**MSE:** Модель 1 имеет MSE равное 0.000137, у модели 2 MSE равно 0.005429. Это означает, что модель 1 в среднем делает более точные прогнозы, чем модель 2.

**MAE:** Модель 1 имеет MAE равное 0.008577, у модели 2 MAE равно 0.035358. Это означает, что в среднем прогнозы модели 1 отклоняются от истинных значений на меньшую величину, чем прогнозы модели 2.

**R^2:** Модель 1 имеет R^2 равное 0.997843, у модели 2 R^2 равно 0.923901. Это означает, что модель 1 лучше объясняет вариацию в данных, чем модель 2.

В целом, по всем этим метрикам модель 1 показывает лучшую производительность, чем модель 2.