1. **Используемые библиотеки и структура файлов**

**NumPy, Pandas** – хранение и обработка массивов информации

**TensorFlow (Keras), TensorflowTransform** – методы машинного обучения

**MatPlotLib, Seaborn**– графики

Файлы с сырыми и предварительно обработанными данными находятся в папке **./Golit/**

Данные в удобном для работы виде в папке **./data/**

Сохранённые модели и веса в папке **./model/**

Весь код находится в файле **pain and suffering.ipynb**

1. **Работа с данными**

**Предварительная обработка данных (10/02/23 найдены новые проблемы и переписано 15/02/23):**

Данные были извлечены из файлов **GOL[year].DAT** со следующими метками (во всех файлах таблица 102):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Year** | **Day number** | **Tmax** | **Tmin** | **Rain** | **Dew** |
| 2006 – 2015 | 2 | 10 | 12 | 19 | 28 |
| 2016 - 2022 | 2 | 12 | 14 | 21 | 34 |

К тому же виду были приведены данные из **GOL[year].csv** (были выгружены из **Golitcino72-17d2.xslx**). Был создан файл с оценками потерь урожая по годам.

Погодные данные за всё время были объединены в один файл и проанализированы при помощи Excel (построен график температуры для всех годов, использована раскраска по величине значения для поиска выбросов), добавлены булевые столбцы:

* Осадки (мм): 1 – осадки >1мм, 0 – остальное
* Роса (доля от суток): 1 – доля росы >0.2, -1 – нет данных, 0 – остальное

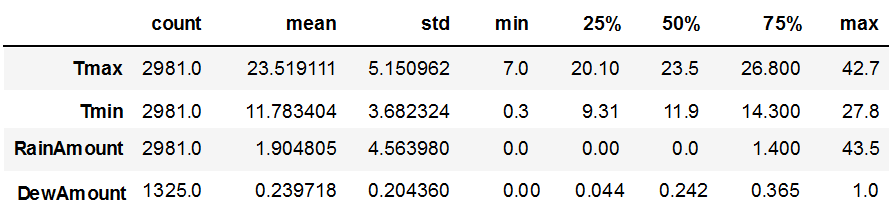
(03/05/23) Осадки: 1 – осадки > 0.1мм, 0 -остальное

**Обучение идёт хуже**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Year** | **Rain** | **Dew** | **Loses** | **Details** |
| 1972 |  |  |  | Скорее всего, датчик осадков был расположен рядом с поливом – наблюдаются аномальные дни с 35-42мм осадков несколько раз в год (>25мм – ливень) |
| 1973 |  |  |  |
| 1977 |  |  |  |
| 1982 |  |  |  |
| 1983 |  |  |  |
| 1984 |  |  |  |
| 1987 |  |  |  |
| 1991 |  |  |  |
| 1992 |  |  |  |  |
| 1993 |  |  |  |
| 1994 |  |  |  |
| 1995 |  |  |  |
| 1996 |  |  |  |
| 1997 |  |  |  |
| 1998 |  |  |  |
| 2002 |  |  |  |
| 2004 |  |  |  |  |
| 2005 |  |  |  |
| 2006 |  |  |  |
| 2007 |  |  |  | В .DAT файле первые 18 дней года |
| 2010 |  |  |  | Аномальная жара, 216-221 нет данных |
| 2011 |  |  |  | Нет данных 222-231 день |
| 2012 |  |  |  |  |
| 2013 |  |  |  |
| 2014 |  |  |  |
| 2015 |  |  |  | Данные с 200 дня аномальные(18 июля) |
| 2016 |  |  |  |  |
| 2017 |  |  |  |
| 2018 |  |  |  | Нет данных о потерях урожая  Эти года были использованы для сравнения моделей между собой |
| 2019 |  |  |  |
| 2020 |  |  |  |
| 2021 |  |  |  |
| 2022 |  | 152 – 195 везде 0 |  |

A picture containing scatter chart

Description automatically generated



**Приведение данных к удобному для обучения формату (20/02/23):**

Все погодные данные были сохранены в файл **GOL\_FULLDATA\_fixed.csv** и далее нормализовывались при помощи Python:

def normalize(array):

ma, mi = max(array), min(array)

res = []

for i in range(len(array)):

res.append((array[i] - mi) / (ma - mi))

return res

Также была учтена зависимость роста числа реинфекций от периода вегетационного сезона (номер дня заменён на индекс периода).

Структура полученного датафрейма, из которого составлялись наборы для обучения:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Year** | **Sprouting** | **Growth** | **Maturation** | **Tmax** | **Tmin** | **Bool\_Rain** | **Bool\_Dew** | **Rain** | **Dew** |
| 1972 - 2022 | T - F | T - F | T - F | [0; 1] | [0; 1] | T - F | T - F | R | {-1} ∪ [0; 1] |

|  |  |
| --- | --- |
| **Year** | **Loses** |
| 1972 - 2022 | [0;1] |

Готовые датафреймы были сохранены в папку .**/data**, чтобы не повторять процедуру предварительной обработки. Сохранение и загрузка:

data.to\_csv('D:\\NIR\\data\\data\_raw\_fixed.csv', index=False)

yield\_loss.to\_csv('D:\\NIR\\data\\loses\_raw.csv', index=False)

# reading preprocessed data from file

data = pd.read\_csv('D:\\NIR\\data\\data\_raw\_fixed.csv')

yield\_loss = pd.read\_csv('D:\\NIR\\data\\loses\_raw.csv')

**Размножение выборки и разделение на training и validation set (22/02/23 + переписано 20/03/23):**

Сначала были выделены несколько лет для валидации, остальные года (с ограничениями) были включены в набор для обучения при помощи **trainval()**

Затем набор для обучения был размножен при помощи функции **replicate()**

**(01/04/23**) Генерация датасетов убрана в функцию **sets\_prepare()**

**Преобразование наборов для RNN (30/04/23):**

Вектор значений разбивается на данные по дням и преобразуется в матрицу при вызове **transform\_for\_RNN()**

**Разбиение данных на группы по потерям урожая (24/04/23, 03/05/23):**

|  |  |  |
| --- | --- | --- |
| **Балл** | **Поражение ботвы** | **Годы** |
| 1 | 0 - 25 | [1972, 1973, 1982, 1987, 1991, 1992, 1993, 1994, 1995, 1997, 1998, 2002, 2005, 2006, 2007, 2009, 2010, 2011, 2014] |
| 2 | 25 – 50 | [1977, 1983, 1984, 1993, 1996, 2008] |
| 3 | 50 – 75 | [1984, 2004, 2012, 2013, 2015, 2016, 2017] |
| 4 | 75 – 100 | - |

|  |  |
| --- | --- |
| **Поражение ботвы** | **Годы** |
| 0 – 10 | [1991, 1994, 1995, 1998, 2006, 2007, 2009, 2014] |
| 10 – 20 | [1977, 1982, 1987, 1993, 1996] |
| 20 – 30 | [1983, 2008] |
| 30 – 40 | [1984] |
| 40 – 50 | [1984, 2015, 2016, 2017] |
| 50 – 60 | [2004, 2012, 2013] |
| 60 – 70 | - |
| 70 – 80 | - |
| 80 – 90 | - |
| 90 – 100 | - |

1. **Сохранение и загрузка из памяти архитектуры модели и весов**

**(6/03/23)** Модель сохраняется в файл формата **TensorFlow SavedModel и содержит архитектуру, конечные веса, конечное состояние оптимизатора, значения метрик:**

model = ...

model.save('path/to/location')

model = keras.models.load\_model('path/to/location')

Сохранение архитектуры отдельно (можно использовать эту же архитектуру чтобы обучиться заново):

Config = model.get\_config()# .to\_json()

Loaded\_model = tf.Sequential.from\_config(config) # tf.Model.from\_config(config)

**(05/05/23)** Все модели сохранены с указанием типа(балльные потери или обучение на всём сразу), вида данных для обучения и видом архитектуры (наличие рекуррентного слоя отмечено)

1. **Метод главных компонент (05/03/23)**

Проблемы с выводом тензора в версии TFv2 – зацикливание??? (ест всю оперативную память)

1. **Глубокое обучение (5/03/23)**

**Входные данные:**

На вход нейронной сети подаётся вектор с данными за некоторое количество дней, на выходе получаем прогнозируемый процент потерь урожая в этом году.

Дни отбираются тремя способами:

1. Первые N дней лета
2. Дни с 4 по 24 июля
3. Случайно или равномерно распределённые по заданному периоду (месяц / 2 месяца / всё лето) дни

Данные по дням содержат:

1. Только температуру
2. Температуру и булевые осадки
3. Температура и осадки
4. Температура, осадки, роса

**Архитектура:**

Для каждого варианта входных данных использовались реккурентная нейронная сеть и многослойный персептрон.

Для вывода информации о структуре после сборки (размерность входа должна быть известна):

model.summary()

**Функции потерь:**

MAE, MSE

**Функции активации (30/04/23):**

ReLu, softsign на выходном слое

**(04/05/23)** в прогнозах для Мичуринска бывают значения -0,01% - заменить на SeLu?

1. **Сравнение моделей**

Оптимальная архитектура:

tf.keras.Sequential([

layers.Dense(21, activation='relu'),

layers.Dense(12, activation='relu'),

layers.Dense(4, activation='relu'),

layers.Dense(1, activation='softsign')

])

**Обучение по группам (24/04/23, 03/05/23):**

Из обучения по опасному периоду исключены 2007, 2010, 2015

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Год** | **По началу лета** | | | | **По опасному периоду** | | | | **Реальные потери** |
| **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** |
| **Голицино** | | | | | | | | | |
| **2018** |  |  |  | - | 11,6 | 32,1 | 60.2 | - | **52,7%** |
|  |  |  | - | 13,4 | 32,2 | 61.7 | - |
| **2019** |  |  |  | - | 9,5 | 26,3 | 54.3 | - | **-** |
|  |  |  | - | 7,7 | 28,7 | 55.5 | - |
| **2020** |  |  |  | - | 10,2 | 28,2 | 56.3 | - | **57,8%** |
|  |  |  | - | 11,2 | 29,5 | 57.7 | - |
| **2021** |  |  |  | - | 11,7 | 33,9 | 60.9 | - | **43,7%** |
|  |  |  | - | 12,1 | 32,4 | 63.3 | - |
| **2022** |  |  |  | - | 12,8 | 31,6 | 58.5 | - | **-** |
|  |  |  | - | 8,6 | 32,7 | 60.8 | - |
| **Мичуринск** | | | | | | | | | |
| **2022** |  |  |  | - | 16,9 | 34,2 | 61.9 | - | **10%** |
|  |  |  | - | 21,7 | 34,7 | 65.4 | - |

**(04/05/23)**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Год** | **По опасному периоду** | | | | | | | | | | **Реальные потери** |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **Голицино** | | | | | | | | | | | |
| **2018** | 8.9 | 17.9 | 30.4 | - | - | - | - | - | - | - | **52,7%** |
| 4.4 | 18.2 | 26.4 | - | - | - | - | - | - | - |
| **2019** | 0 | 13.7 | 23.5 | - | - | - | - | - | - | - | **-** |
| 1.5 | 9.5 | 21.4 | - | - | - | - | - | - | - |
| **2020** | 0 | 12.3 | 28.6 | - | - | - | - | - | - | - | **57,8%** |
| 4.4 | 15.5 | 17.7 | - | - | - | - | - | - | - |
| **2021** | 6.7 | 18.3 | 31.5 | - | - | - | - | - | - | - | **43,7%** |
| 9.8 | 17.7 | 18.9 | - | - | - | - | - | - | - |
| **2022** | 9.5 | 13.1 | 30.2 | - | - | - | - | - | - | - | **-** |
| 1.4 | 17.1 | 22.4 | - | - | - | - | - | - | - |
| **Мичуринск** | | | | | | | | | | | |
| **2022** | 6.4 | 19.7 | 35.4 | - | - | - | - | - | - | - | **10%** |
| 2.4 | 35.1 | 27.7 | - | - | - | - | - | - | - |