

ГБУ ДО «Региональный центр выявления, поддержки и развития способностей и талантов у детей и молодежи Челябинской области «Курчатов Центр».

Всероссийский конкурс юных инженеров-исследователей с международным участием «Спутник»

**ПРОГНОЗИРОВАНИЕ АСТРОНОМИЧЕСКИХ ВЕЛИЧИН НА ОСНОВЕ
МЕТОДА РЕГРЕССИИ В МАШИННОМ ОБУЧЕНИИ.**

«Информационные технологии»

Выполнил: Даниленко В.О.

Класс: 8

Место жительства: г. Челябинск,

Челябинская область

Научный руководитель: преподаватель

Верховских Игорь Вячеславович

Самара, 2023

Оглавление

Аннотация.	3
Введение.	4
1. Анализ существующих решений	5
1.1 Диаграмма Герцшпрунга — Рассела	5
1.2 Применение ИИ в астрономии.	6
2. Пошаговая реализация проекта.	7
2.1 Разработка регрессионной модели в Jupyter Notebook.	7
2.2 Создание приложения FLASK в Visual Studio Code.	10
2.3 Запуск FLASK приложения.	14
2.4 Создание удаленного репозитория.	15
Описание результата проекта.....	16
Список использованных источников.....	17
Приложение А.....	18
Приложение Б	19

Аннотация.

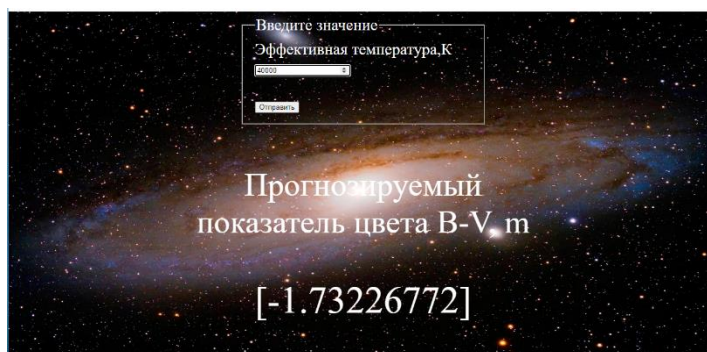
В астрономии очень много косвенно зависимых величин, между которыми нельзя однозначно поставить знака равенства. Многие зависимости имеют приблизительный, оценочный характер. Слишком много факторов нужно учитывать для оценки той или иной величины, прибегать к сложнейшим расчетам, чтобы выявить те или иные зависимости. Почему бы нам не применить методы машинного обучения и не спрогнозировать на основании определенного набора данных искомую величину. Пример таких зависимостей можно рассмотреть на основе диаграммы Герцшпрунга-Рассела.

Мы рассмотрим каким образом обучить модель и вывести ее в продакшен при помощи Фреймворка Flask. В Jupyter Notebook обучим простую регрессионную модель, создадим синтетический датасет на основе таблицы (Приложение А), мы будем прогнозировать Показатель цвета $B-V$,^m в зависимости от Эффективной температуры, К.

Для того, чтобы получить точную модель для прогнозирования астрономической величины необходимо проанализировать множество методов, алгоритмов регрессионного анализа, сравнить их на основании оценки таких показателей как: MAE (средняя абсолютная ошибка), R^2 (коэффициент детерминации) и т.д., а затем выбрать наиболее лучшие. Этот процесс имеет достаточно большие временные затраты, поэтому мы остановимся пока на самой первой ступени и применим простую регрессионную модель.

После этого перейдем в программу Visual Studio Code. Создадим директорию MY_FLASK. Добавим в структуру модель lr_model.pkl . Для запуска приложения Flask создадим файл app.py и main.

При переходе в браузер открывается html страница нашего приложения. Таким образом выглядит html страница нашего Flask приложения.



Итак, в открывшемся окне пользователю необходимо ввести значение эффективной температуры в Кельвинах и нажать на кнопку: отправить. На выходе пользователь получает значение Прогнозируемого показателя цвета $B-V$,^m. Наше приложение работает успешно.

Введение.

Темой моей работы является «Прогнозирование астрономических величин на основе метода регрессии в машинном обучении.»

Актуальность: На сегодняшний день данные полученные человеком о космосе растут с геометрической прогрессией, а, следовательно, возникает все большая потребность в аналитике данной информации при помощи ИИ, применении машинного обучения.

Гипотезой нашей работы является утверждение, что задачи регрессии применимы в области астрономии, в частности мы можем прогнозировать показатель цвета $B-V,^m$, в зависимости от эффективной температуры, K .

Объектами исследования является процесс прогнозирования астрономических величин, в частности показателя цвета $B-V,^m$.

Предмет исследования —автоматизация процесса прогнозирования астрономических величин, в частности показателя цвета $B-V,^m$.

Цель работы: написать FLASK приложение, которое будет выдавать прогнозное значение величины в зависимости от эффективной температуры, K .

Задачи исследования: изучить понятия: диаграмма Герцшпрунга — Рассела, эффективная температура, показатель цвета $B-V$, регрессия в машинном обучении; разработать регрессионную модель в Jupyter Notebook; создать файл app.py для приложения FLASK в Visual Studio Code; создать файл web страницы для приложения FLASK в Visual Studio Code.

1. Анализ существующих решений

1.1 Диаграмма Герцшпрунга — Рассела

Диаграмма Герцшпрунга — Рассела (Рессела, сокращённо диаграмма Г—Р) — диаграмма рассеяния, используемая в астрономии, которая представляет зависимость между абсолютной звёздной величиной и спектральным классом для звёзд, либо между другими величинами, которые тесно связаны с этими параметрами. В любом случае, в верхней части диаграммы оказываются яркие звёзды, а в нижней части — тусклые; в левой части — горячие звёзды голубого цвета, в правой — холодные и красные. В качестве синонимов основному термину также используются понятия «**диаграмма спектр — светимость**», «**диаграмма светимость — эффективная температура**» и другие, хотя, более строго, различные названия относятся к определённым вариантам диаграммы [1].

Диаграмма (рисунок 1) названа в честь Эйнара Герцшпрунга и Генри Норриса Расселла, которые впервые её построили в разных вариантах в 1911 и 1913 годах.

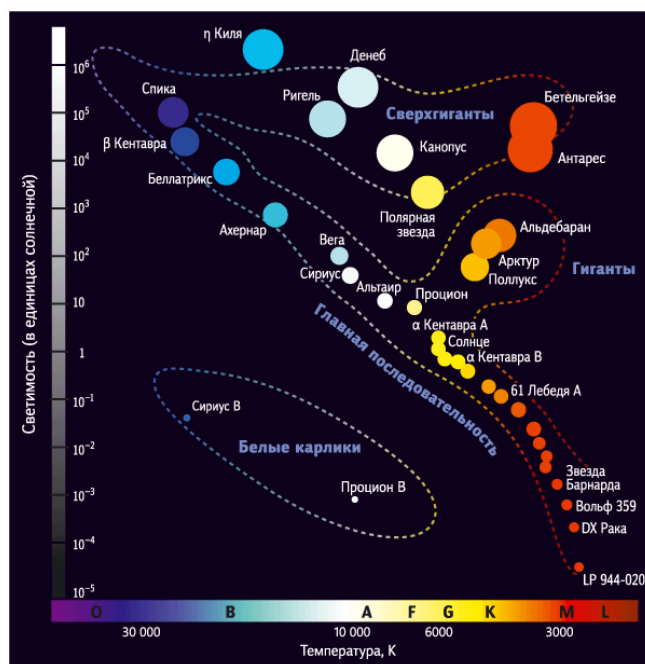


Рисунок 1.- Диаграмма Герцшпрунга — Рассела

Эффективная температура — параметр, характеризующий светимость (полную мощность излучения) небесного тела (или другого объекта), то есть это температура абсолютно чёрного тела с размерами, равными размерам небесного тела и излучающего такое же количество энергии в единицу времени [2].

Показатель цвета $B-V$ (« B минус V ») — один из двух показателей цвета фотометрической системы UBV . Наиболее широко используемая характеристика цвета астрономических объектов. Как и другие показатели цвета, $B-V$ характеризует распределение энергии в спектре объекта, то есть его цвет [3].

По показателю цвета звезды можно сделать примерные выводы о её температуре. Чем больше показатель цвета, тем холоднее звезда (и тем более поздний её спектральный класс). Если звезда излучает как абсолютно чёрное тело с температурой T , то связь между показателем цвета и температурой имеет вид (1):

$$T = 4600 \left(\frac{1}{0,92(B - V) + 1,7} + \frac{1}{0,92(B - V) + 0,62} \right) \quad (1)$$

В действительности на цвет звёзд влияет не только температура, но и другие факторы, в частности, химический состав — например, у углеродных звёзд. Поэтому приведённая зависимость является лишь приближённой. Для холодных звёзд она соблюдается хуже, чем для горячих. Построению эмпирической и полуэмпирической зависимости между температурой и показателем цвета посвящена обширная литература [3].

1.2 Применение ИИ в астрономии.

Искусственный интеллект (ИИ) играет все более важную роль в астрономии, помогая ученым анализировать огромные объемы данных, делать новые открытия и понимать Вселенную глубже[4]. В этом разделе мы рассмотрим, как ИИ применяется в астрономии и какие преимущества он может принести (Таблица 1).

Таблица 1-Применение искусственного интеллекта в астрономии.

Тема	Описание	Примеры
Анализ космических данных	Применение алгоритмов машинного обучения для обработки и анализа больших объемов данных, полученных из космических наблюдений.	Классификация галактик, определение свойств звезд, поиск новых объектов в космосе.
Поиск экзопланет	Использование искусственного интеллекта для обнаружения и классификации планет вне Солнечной системы.	Автоматическое обнаружение периодических изменений яркости звезд, анализ спектров для определения наличия планетных атмосфер.
Автоматическое распознавание объектов	Применение нейронных сетей и компьютерного зрения для автоматического распознавания и классификации объектов в космосе.	Распознавание галактик, звездных скоплений, астероидов и комет на основе изображений.
Моделирование и прогнозирование	Использование искусственного интеллекта для создания моделей и прогнозирования различных астрономических явлений.	Прогнозирование солнечных вспышек, моделирование эволюции галактик, предсказание траекторий астероидов.

Около трех лет назад стала популярной задача поиска экзопланет на основе данных, полученных с космической обсерватории НАСА Кеплер <https://exoplanetarchive.ipac.caltech.edu/docs/data.html>, с использованием ИИ. Так же были успешно применены сверточные нейронные сети для решения транзитных задач поиска экзопланет. Я хотел прописать модель для данной задачи, и предоставить ее на конкурс, но т.к. требования были направлены на более прикладной характер, то решил разработать FLASK приложение на основе регрессионной модели. Регрессия в машинном обучении - задача контролируемого машинного обучения, которая прогнозирует значение метки по набору связанных компонентов. Метка здесь может принимать любое значение, а не просто выбирается из конечного набора значений, как в задачах классификации. Алгоритмы регрессии моделируют зависимость меток от связанных компонентов, чтобы определить закономерности изменения меток при разных значениях компонентов.

2. Пошаговая реализация проекта.

2.1 Разработка регрессионной модели в Jupyter Notebook.

Мы рассмотрим каким образом обучить модель и вывести ее в продакшен при помощи Фреймворка Flask. В Jupyter Notebook обучим простую регрессионную модель, создадим синтетический датасет на основе таблицы (Приложение А), мы будем прогнозировать Показатель цвета $B-V$,^m в зависимости от Эффективной температуры, K .

После сохранения модели создадим приложение с использованием Фреймворка Flask. А также попробуем вывести его в продакшен, разместив

Импортируем библиотеки (рисунок 2): Pandas, Numpy, Модель линейной регрессии, так же импортируем метод `train_test_split` для разделения выборки на обучающую и тестовую. Так же потребуется библиотека `Pickle`, благодаря которой мы сможем сохранять нашу модель [5,6,7].

```
B [6]: import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

import pickle
```

Рисунок 2- Часть кода в Jupyter Notebook.

Создадим синтетический датасет (рисунок 2), предположим у нас есть прогнозировать показатель цвета $B-V$,^m обозначим как bv и Эффективная температура, K обозначим как k Это будут простые Питоновские списки, данные можно заполнять произвольно, это большой роли не играет. Сделаем по 14 элементов согласно таблице Приложение А.

```
]: bv=[-0.35,-0.31,-0.17,0.0,0.16,0.30,0.45,0.57,0.70,0.84,1.11,1.39,1.61,2.00]
   k=[40000,28000,15500,10000,8500,7400,6600,6600,5400,4700,4000,3600,3000,2660]
```

Рисунок 3- Часть кода в Jupyter Notebook.

Проверим размерность нашего списка (рисунок 4):

```
print(len(bv))
print(len(k))
```

```
14
14
```

Рисунок 4- Часть кода в Jupyter Notebook.

Размерности совпадают.

Разделим данные (рисунок 5) на X и Y, где X соответственно эффективная температура **k**, а Y прогнозировать показатель цвета B–V **bv**. Сразу создадим массив `np.array` и передадим туда эффективную температуру **k**, сделаем `reshape`, чтобы у нас не было конфликта размерности. Также создадим массив `np.array` и передадим туда показатель цвета B–V **bv**.

```
x=np.array(k).reshape(-1,1)
y=np.array(bv)
```

```
train_test_split
```

Рисунок 5- Часть кода в Jupyter Notebook.

После этого разделим наши X и Y на обучающую и проверочную выборку (рисунок 6). Передаем `X_train`, `X_test`, `Y_train`, `Y_test`, воспользуемся методом `train_test_split` в качестве аргументов передаем наши X,Y, также передаем параметр перемешивания `shuffle`, так же передаем `test_size` у нас будет 20% на валидацию.

```
x_train,x_test,y_train,y_test=train_test_split(x, y,test_size=0.2,shuffle=True,random_state=42)
```

Рисунок 6- Часть кода в Jupyter Notebook.

Обучим наш алгоритм линейной регрессии (рисунок 7). Создадим переменную `lr` и поместим в нее наш алгоритм, обучим на `train` выборке. И создадим переменную `y_pred` у нас там будут наши прогнозные значения, которые мы передаем при помощи метода `predict` на обученной модели. И передаем туда нашу тестовую выборку `X_test`.

ML

```
#linear regression
lr=LinearRegression()
lr.fit(x_train,y_train)
y_pred=lr.predict(x_test)
```

Рисунок 7- Часть кода в Jupyter Notebook.

Посмотрим, что у нас передалось в `y_pred` (рисунок 8). Выведем прогнозный показатель цвета B-V **bv** и соответствующий ему `X_test` значения эффективной температуры **k**

```
y_pred
array([ 0.89455227,  0.97640785, -1.73226772])

x_test
array([[ 4700],
       [ 3600],
       [40000]])
```

Рисунок 8- Часть кода в Jupyter Notebook.

Сохраним нашу модель (рисунок 9). Воспользуемся методом `pickle.dump`, передадим туда в качестве аргумента наш алгоритм обученный, откроем `lr_model.pkl` в формате записи `wb`. После этого модель появляется в памяти ноутбука.

сохраняем модель

```
pickle.dump(lr,open('lr_model.pkl','wb'))
```

Рисунок 9- Часть кода в Jupyter Notebook.

Откроем сохраненную модель в режиме чтения `rb` (рисунок 10)

```
model_load=pickle.load(open('lr_model.pkl','rb'))
```

Рисунок 10- Часть кода в Jupyter Notebook.

Воспользуемся методом `predict` для `model_load` (рисунок 11).

```
model_load.predict(x_test)
array([ 0.89455227,  0.97640785, -1.73226772])
```

Рисунок 11- Часть кода в Jupyter Notebook.

Нам необходимо убедиться, что `y_pred` и `model_load.predict` выдают одинаковые значения. Мы видим, что прогнозные значения изначально обученной модели совпадают с

прогноznыми значениями нашей загруженной модели из файла. Это значит, что наш файл с моделью `lr_model.pkl` мы можем использовать в приложении с использованием Фреймворка Flask. Для того, чтобы это сделать нужно скачать модель на наш локальный компьютер.

2.2 Создание приложения FLASK в Visual Studio Code.

После этого перейдем в программу Visual Studio Code. Создадим директорию `MY_FLASK` (рисунок 12). Добавим в структуру модель `lr_model.pkl`. Для запуска приложения Flask создадим файл `app.py`.

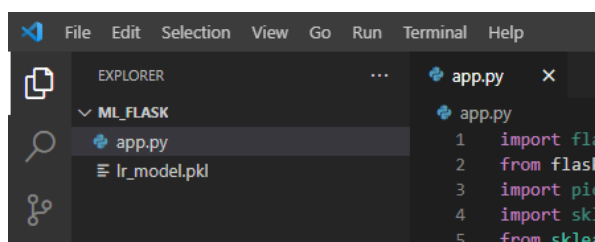


Рисунок 12- Часть кода в Visual Studio Code.

Импортируем библиотеку Flask для того, чтобы работать с данным Фреймворком. После этого также импортируем метод `render_template`, который находится в библиотеке Flask, он позволяет взаимодействовать Фреймворку с html разметками страниц и при необходимости взаимодействовать с web формами. Импортируем библиотеку `pickle` для того, чтобы загрузить обученную модель `lr_model.pkl` из нашей структуры проекта, и мы могли к ней обратиться. Импортируем библиотеку `sklearn`, а также линейную регрессию (рисунок 13).

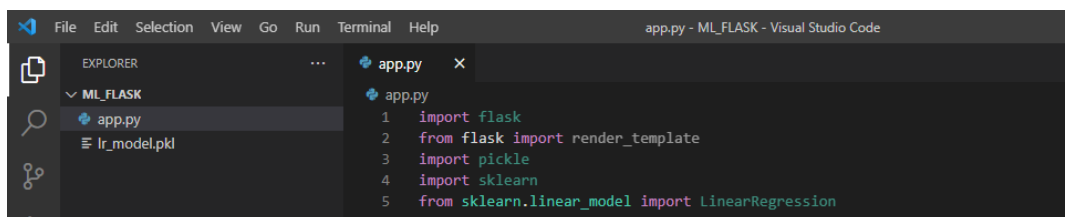


Рисунок 13- Часть кода в Visual Studio Code.

Попробуем запустить приветственное сообщение «Hello flask» (рисунок 14).

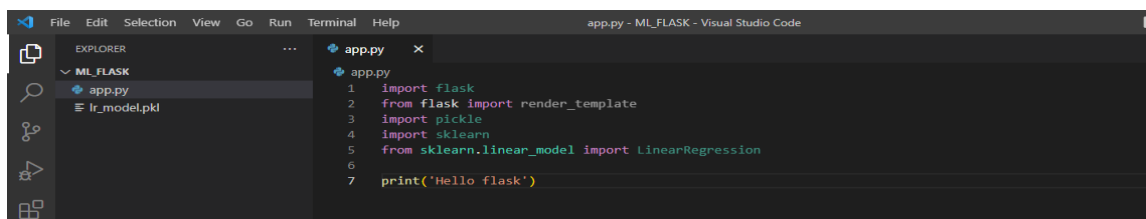


Рисунок 14- Часть кода в Visual Studio Code.

При запуске `app.py` получаем ошибки, связанные с импортом библиотек. Поэтому в командной строке при помощи `pip install` установим все необходимые библиотеки:

- pip install flask
- pip install pickle-mixin
- pip install sklearn
- pip install gunicorn
- pip install nampy

Библиотека gunicorn поможет инициализировать приложение с помощью Pandas.

После установки библиотек, возвращаемся к тесту нашего приложения, выводим на печать «Hello flask». Приложение отработало наш скрипт и вывело на печать требуемую фразу. Библиотеки загружены успешно, удаляем тестовую строку[8].

Продолжим писать наше приложение, для начала мы его инициализируем. Вызываем метод Flask, пишем метод name и указываем директорию в которой у нас будут храниться html шаблоны, назовем его 'templates'. Сразу создадим директорию templates (рисунок 15).

```
app=flask.Flask(__name__, template_folder='templates')
```

Рисунок 15- Часть кода в Visual Studio Code.

После этого создаем декораторы, которые будут соответствовать адресам наших страниц на нашем приложении, так же указываем методы, с которыми будут работать наши страницы: POST, GET. Начальная страница у нас будет пустая, а страничка index также будет работать с методами POST, GET. Соответственно, когда мы используем метод POST мы отправляем информацию, когда используем метод GET мы получаем информацию (рисунок 16).

```
@app.route('/', methods=['POST', 'GET'])
@app.route('/index', methods=['POST', 'GET'])
```

Рисунок 16- Часть кода в Visual Studio Code.

На нашей странице мы пишем функцию def main(), которая будет обрабатывать web форму нашей страницы, получать оттуда значения эффективной температуры, K и возвращать показатель цвета $B-V^m$

Пользуемся методом request, если у нас метод GET, то мы ничего не делаем просто возвращаем шаблон. Наш шаблон будет называться main.html. Создадим main.html. Если мы используем метод POST, значит у нас передана в наше приложение какая-то информация с эффективной температурой, K. Тогда мы открываем нашу модель lr_model.pkl в режиме чтения rb, передаем в переменную f и создаем переменную с нашей загруженной моделью loaded_model, это будет pickle.load передаем туда переменную f ,в

которую загружен файл [9]. Создаем переменную exp, это будет число с плавающей точкой, передаем туда наш опыт, который поступает с web формы. Создаем прогнозный Показатель цвета B-V, m - y_pred. Возвращаем так же наш шаблон main.html и так же возвращаем результат y_pred (рисунок 17).

```
def main():
    if flask.request.method=='GET':
        return render_template('main.html')

    if flask.request.method=='POST':
        with open('lr_model.pkl','rb') as f:
            loaded_model = pickle.load(f)

        exp = float(flask.request.form['experience'])
        y_pred = loaded_model.predict([[exp]])

        return render_template('main.html',result = y_pred)
```

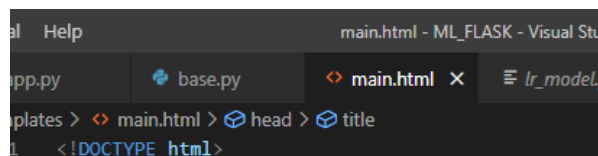
Рисунок 17- Часть кода в Visual Studio Code.

На этом приложение наше закончено, инициализируем его и сохраняем (рисунок 18).

```
if __name__ == "__main__":
    app.run()
```

Рисунок 18- Часть кода в Visual Studio Code.

Теперь нам необходимо написать код для html странички [10]. Для этого нам потребуется гипертекстовая разметка html. Инициализируем doctype и говорим, что у нас это html (рисунок 19).



```
<!DOCTYPE html>
```

Рисунок 19- Часть кода в Visual Studio Code.

Далее указываем какой у нас будет стиль. Создаем контейнер для этого, передаем туда нашу web форму, передаем туда параметры: внешний отступ на всех четырёх сторонах элемента у нас будет авто, размер, указываем в процентах 35%, цвет и размер шрифта (рисунок 20).

```
<style>
form{
    margin: auto;
    width: 35%;
    color: #fff;
    font-size: 30px;
}
```

Рисунок 20- Часть кода в Visual Studio Code.

И форму для результата: внешний отступ на всех четырёх сторонах элемента у нас будет авто, размер, указываем в процентах 50% (рисунок 21).

```
.result{
    margin: auto;
    width: 50%;
}

</style>
```

Рисунок 21- Часть кода в Visual Studio Code.

Теперь мы создадим контейнер для форм, куда мы будем передавать значение соответственно эффективной температуры, К. Главный тег Head, указывает, что это у нас заголовок. У заголовка будет название, тег title: Прогнозирование показателя цвета B-V,^m (рисунок 22).

```
<head>
  <title>
    Прогнозирование показателя цвета B-V, m
  </title>
</head>
```

Рисунок 22- Часть кода в Visual Studio Code.

И описываем теперь нашу web форму. Вводим команду action, и она должна у нас взаимодействовать с нашим приложением. И указываем, что работает метод POST, который мы указали в flask приложении. В теле пишем, что мы ожидаем туда ввода значений, для этого указываем тег fieldset. В теле указываем легенду -Введите значение-Эффективная температура, К-. Далее вводим значение, то, что указали во flask как experience указываем тип число. И у нас это обязательно. Сделаем несколько отступов с красной строки, тег br. И создаем кнопку отправить, это тип называется submit (рисунок 23).

```
<form action="{{url_for('main')}}" method="POST">
  <fieldset>
    <legend>Введите значение</legend>
    Эффективная температура, К
    <input name = "experience" type="number"
      required>
    <br>
    <br>
    <input type="submit">
  </fieldset>
```

Рисунок 23- Часть кода в Visual Studio Code.

Создадим контейнер, куда мы передаем результаты. Указываем, что возвращаем result из приложения Flask, выравниваем его по центру. Проверяем условия, если у нас есть результат делаем отступ. Открываем тег, передаем туда цвет, размер шрифта, и туда передаем –Прогнозируемый показатель цвета B-V,^m. И нам нужно вернуть переменную result. Так же укажем размер, цвет шрифта и передадим туда наш результат (рисунок 24).

```

</form>

<div class="result" align="center">
  {% if result %}
  <br>
  <p style="font-size:60px;color:■ #fff ">
    Прогнозируемый показатель цвета В-V, м
  </p>
  <p style="font-size:70px;color:■ #fff "> {{result}} </p>
  {% endif %}
</div>

```

Рисунок 24- Часть кода в Visual Studio Code.

Создадим контейнер для фонового изображения, туда передадим url нашего изображения и размер (рисунок 25).

```

<style type="text/css">
  body{
    background-image: url(https://starcatalog.ru/images/2020/09/m31-3-1.jpg);
    background-size: 100% 100%;
    background-repeat: no-repeat;
  }
</style>

```

Рисунок 25- Часть кода в Visual Studio Code.

2.3 Запуск FLASK приложения.

Запустим наше Flask приложение app.py, в минимальном его наборе [11,12]. В качестве результата у нас появляется наше приложение на нашем локальном хосте. При запуске app.py появляется адрес <http://127.0.0.1:5000> , и мы можем его открыть на нашем браузере (рисунок 26).

```

PS C:\Users\d8978\OneDrive\Рабочий стол\Мама проект\ML_FLASK> & C:\Users\d8978\AppData\Local\Programs\Python\Python310\python.exe "c
:/Users/d8978/OneDrive/Рабочий стол/Мама проект/ML_FLASK/app.py"
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000

```

Рисунок 26- Часть кода в Visual Studio Code.

При переходе в браузер открывается html страница нашего приложения. Таким образом выглядит html страница нашего Flask приложения (рисунок 27).

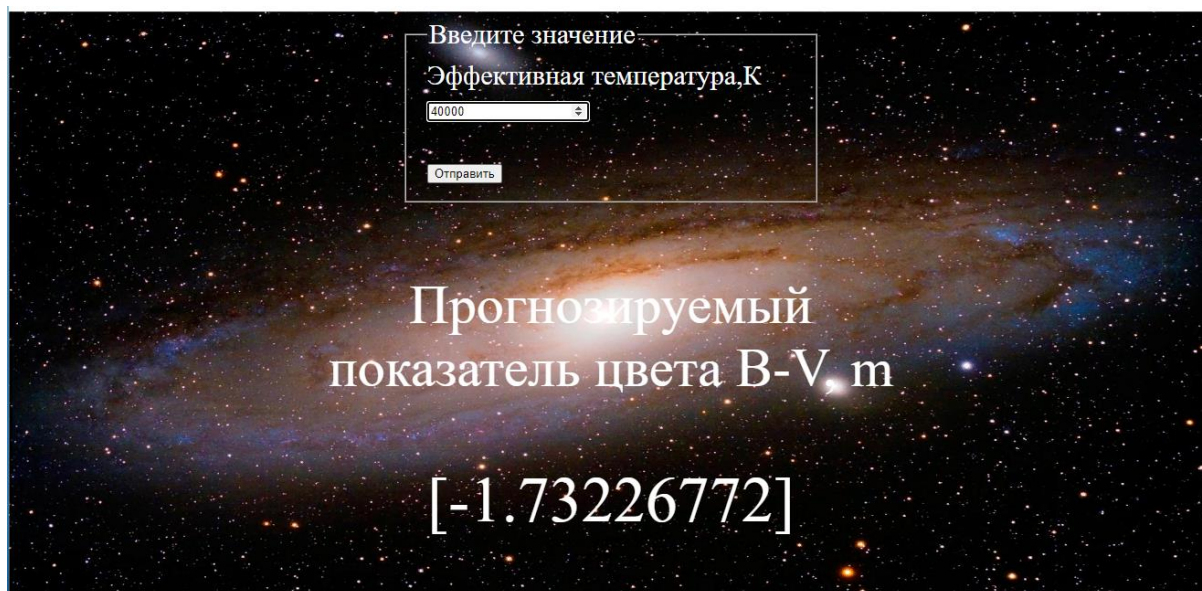


Рисунок 27- HTML страница приложения FLASK.

Итак, в открывшемся окне пользователю необходимо ввести значение эффективной температуры в Кельвинах и нажать на кнопку: отправить. На выходе пользователь получает значение Прогнозируемого показателя цвета $B-V, m$. Наше приложение работает успешно.

2.4 Создание удаленного репозитория.

Репозиторий создали на github.com по адресу: <https://github.com/VOVADAN/-B-V-m>

Описание результата проекта:

Далее я планирую выложить данное Flask приложение в интернете, чтобы любой желающий мог получить к нему доступ. Для этого хотел использовать бесплатный хостинг Heroku, но в данный момент на нем не доступна регистрация для пользователей из России. Поэтому планирую воспользоваться back4app (Приложение Б). Где мы создаем контейнер (бесплатно можно создать 1 контейнер) и привязываем его к нашему репозитарию на github.com.

Для корректного развертывания нашего приложения мы должны прописать Dockerfile, где прописан порядок взаимодействия со всеми компонентами нашего приложения FLASK и require.txt, где прописаны требования к библиотекам. Данные файлы написаны и выложены на github.com. Нам остается только решить вопрос времени для отладки развертывания), но и конечно же разрешить конфликт, связанный с использованием локального сервера Flask. Для развертывания приложения в рабочей среде нам нужен другой сервер и одним из вариантов является использование сервера WSGI.

```
Debug mode: ON  
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.  
* Running on http://127.0.0.1:5000
```

Нам необходимо пересмотреть наш файл App.py, но данным вопросом займемся в будущем, а пока на github.com выложен исходный файл App.py, который запускает нашу программу на локальном сервере Flask.

Так же можно сделать вывод, о том, что наша регрессионная модель не совсем точная. Это может быть связано, с тем, что она обучалась на небольшом наборе данных, всего 14 элементов. Так же мы можем применить и другие алгоритмы регрессионного анализа, в основе которых лежат иные методы расчетов, сравнить их и подобрать наиболее точные. Однозначно наша регрессионная модель требует доработки, над чем в дальнейшем мы будем работать, особенно надо проработать вопрос выбора функции описывающей модель.

Но наше приложение дает нам возможность увидеть показатель цвета $B-V^m$ при любом значении эффективной температуры, по всей температурной шкале.

Я считаю, что применение регрессионного анализа в астрономии позволит выявить еще больше новых закономерностей, основываясь на обработке растущей с геометрической прогрессией базе знаний человека о космосе.

Список использованных источников

1. Википедия [Электронный ресурс]. - Режим доступа:
https://ru.wikipedia.org/wiki/Диаграмма_Герцшпрунга_—_Рассела
2. Бычков К. В. Эффективная температура // Физическая энциклопедия : [в 5 т.] / Гл. ред. А. М. Прохоров. — М.: Большая российская энциклопедия, 1999. — Т. 5: Стробоскопические приборы — Яркость. — С. 645—646. — 692 с. — 20 000 экз. — [ISBN 5-85270-101-7](#).
3. Википедия [Электронный ресурс]. - Режим доступа:
https://ru.wikipedia.org/wiki/Показатель_цвета_B-V.
4. Искусственный интеллект в астрономии: применение и перспективы [Электронный ресурс]. - Режим доступа: <https://nauchniestati.ru/spravka/iskusstvennyj-intellekt-v-astronomii-analiz-kosmicheskikh-dannyh-i-poisk-ekzoplanet/#author>.
5. Документация по библиотеке numpy [Электронный ресурс]. - Режим доступа:
<https://numpy.org/doc/1.22/user/index.html#user>.
6. Документация по библиотеке pandas [Электронный ресурс]. - Режим доступа:
https://pandas.pydata.org/docs/user_guide/index.html#user-guide.
7. Документация по языку программирования python [Электронный ресурс]. - Режим доступа: <https://docs.python.org/3.8/index.html>.
8. Плас Дж. Вандер, Python для сложных задач: наука о данных и машинное обучение. Санкт-Петербург: Питер, 2018, 576 с.
9. Миронов А.А. Машинное обучение часть I ст.9 [Электронный ресурс]. - Режим доступа: <http://is.ifmo.ru/verification/machine-learning-mironov.pdf>.
10. Роббинс, Дженнифер. HTML5: карманный справочник, 5-е издание.: Пер. с англ. - М.: ООО «И.Д. Вильямс»: 2015. - 192 с.: ил.
11. Руководство по быстрому старту в flask [Электронный ресурс]. - Режим доступа:
<https://flask-russian-docs.readthedocs.io/ru/latest/quickstart.html>
12. Силен Дэви, Мейсман Арно, Али Мохамед. Основы Data Science и Big Data. Python и наука о данных. – СПб.: Питер, 2017. – 336 с.: ил.

Таблица - Связь между спектральным классом, эффективной температурой и показателем цвета B–V.

Спектральный класс	Эффективная температура, К	Показатель цвета B–V, m
O5	40000	–0,35
B0	28000	–0,31
B5	15500	–0,17
A0	10000	0,0
A5	8500	0,16
F0	7400	0,30
F5	6600	0,45
G0	6600	0,57
G5	5400	0,70
K0	4700	0,84
K5	4000	1,11
M0	3600	1,39
M5	3000	1,61
M8	2660	2,00

Серверная платформа Back4App .

