НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

Курсовий проект

із дисципліни «Математичне моделювання»

на тему

«Predicting of CO2 emissions»

Виконав:

студент групи КМ-81

Піткевич І.В.

Керівник:

Норкін Б. В.

Київ 2021

Зміст

Використані технології

В даній курсовій роботі було мову програмування Python та її бібліотека, а саме:

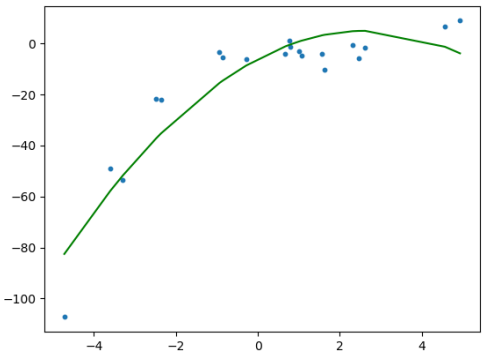
- Sklearn
- Streamlit
- Pandas
- Numpy
- Seaborn
- Matplotlib
- Plotly

Теоретичні відомості

Регресія визначається як метод пошуку зв’язку між незалежними та залежними змінними для прогнозування результату. Перша поліноміальна регресійна модель була використана в 1815 році Гергонною. Він використовується для пошуку найкращої лінії підходу, використовуючи лінію регресії для прогнозування результатів. Існує багато видів регресійної техніки, поліноміальна регресія - одна з них. Перш ніж зрозуміти це, доцільно мати належні знання про лінійну регресію, тому легко буде позначити відмінності між ними. Основна відмінність між лінійною та поліноміальною регресією полягає в тому, що лінійна регресія вимагає лінійно пов'язаних залежних та незалежних змінних, в той час як це може краще відповідати лінії, якщо ми включимо в рівняння будь-який вищий ступінь до незалежного терміна змінної. Рівняння поліноміальної регресії, що має n-й ступінь, можна записати так:

$$Y = b0 + a1x + a2x \wedge 2 + a3x \wedge 3 + \ldots. anx \wedge n$$

Якщо ми додамо більш високі ступені, такі як квадратична, то це перетворює лінію в криву, яка краще відповідає даних. Як правило, він використовується, коли точки в наборі даних розкидані, а лінійна модель не в змозі чітко описати результат. Ми завжди слідкуємо за переоснащенням та недостатністю, враховуючи ці ступені до рівняння. Краще врахувати ступінь, який проходить через усі точки даних, але іноді з більш високим ступенем, таким як 10 або 20, може пройти через усі точки даних і зменшити помилку, але він також фіксує шум даних, що відповідає моді та цього можна уникнути, додавши в набір навчальних даних більше зразків. Отже, завжди бажано вибирати оптимальний ступінь, щоб відповідати моделі. Наприклад:



Опис обраного датасету

Для прогнозування на основі поліноміальної регресії потрібно визначитися з початковими умовами.
Згідно з постановкою задачі, ми маємо датасет під назвою “FuelConsumption.csv”. Він містить рейтинги споживання палива для конкретної моделі та оцінки викиду вуглекислого газу для нових легкових автомобілів для роздрібного продажу в Канаді. Джерело набору даних.
Далі буде описана структура цього csv файлу:

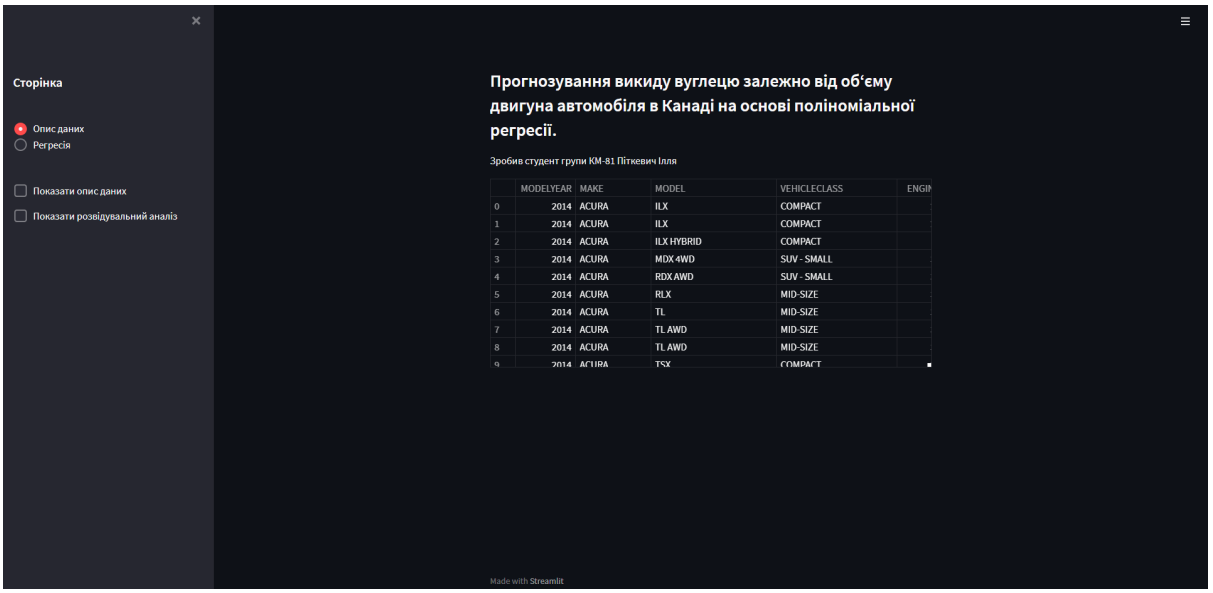
- **MODELYEAR** (рік, в якому випущена автівка), наприклад 2013
- **MAKE** (марка) наприклад Toyota
- **MODEL** (модель), наприклад HILUX
- **VEHICLE CLASS** (тип автівки), наприклад кроссовер
- **ENGINE SIZE** (об‘єм двигуна), наприклад 4.2
- **CYLINDERS** (кількість циліндрів), наприклад 6
- **TRANSMISSION** (трансміссія), наприклад A6
- **FUEL CONSUMPTION in CITY(L/100 km)** (споживання топлива в місті на 100 км)
- **FUEL CONSUMPTION in HWY (L/100 km)** (споживання топлива на автомагістралі на 100 км)
- **FUEL CONSUMPTION COMB (L/100 km)**(споживання топлива комбіноване на 100 км)
- **CO2 EMISSIONS (g/km)** (викид вуглецю грам/км)

Дані були перевірені на якість, тобто відсутність значень або дублікати.

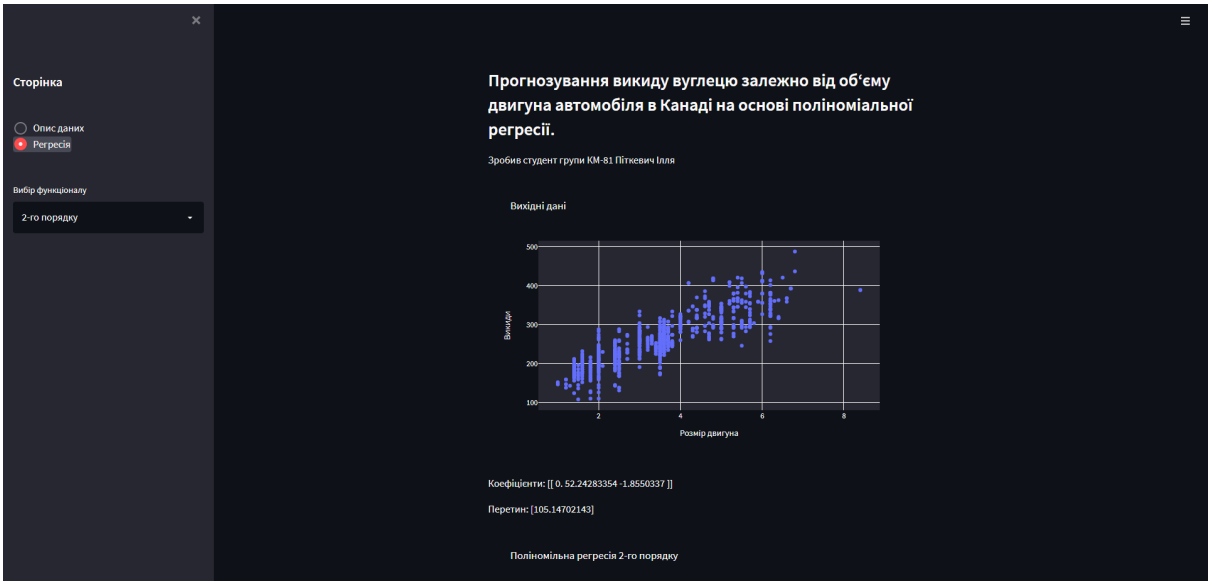
```
MODELYEAR                 0.0
MAKE                      0.0
MODEL                     0.0
VEHICLECLASS              0.0
ENGINESIZE                0.0
CYLINDERS                 0.0
TRANSMISSION              0.0
FUELTYPE                  0.0
FUELCONSUMPTION_CITY      0.0
FUELCONSUMPTION_HWY       0.0
FUELCONSUMPTION_COMB      0.0
FUELCONSUMPTION_COMB_MPG  0.0
CO2EMISSIONS              0.0
dtype: float64
```

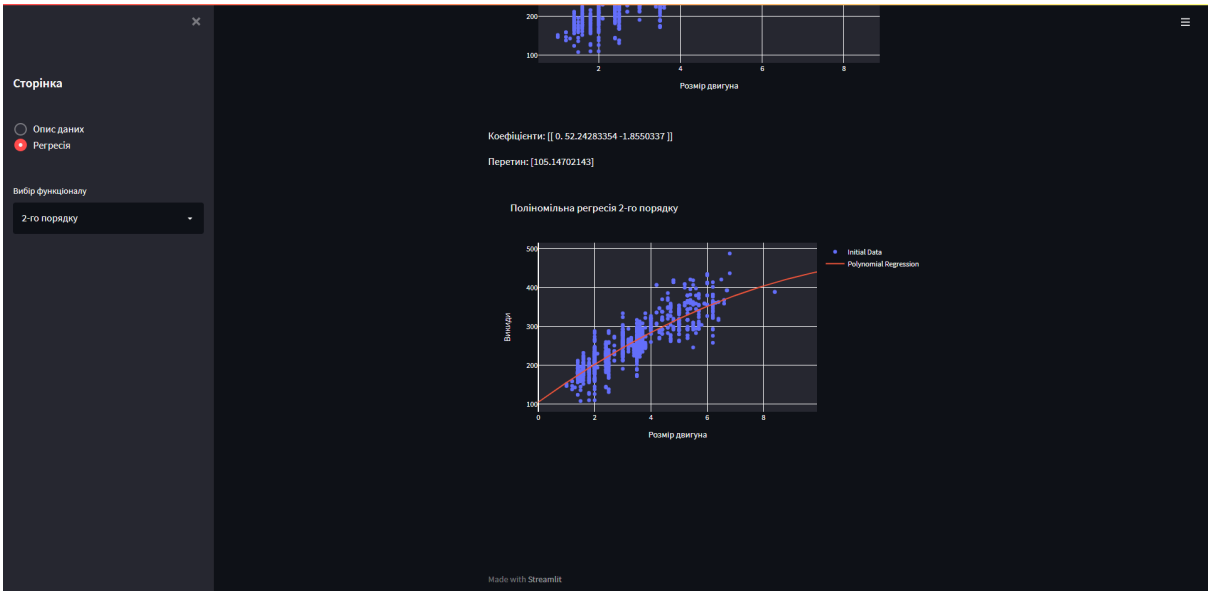
Опис проекту

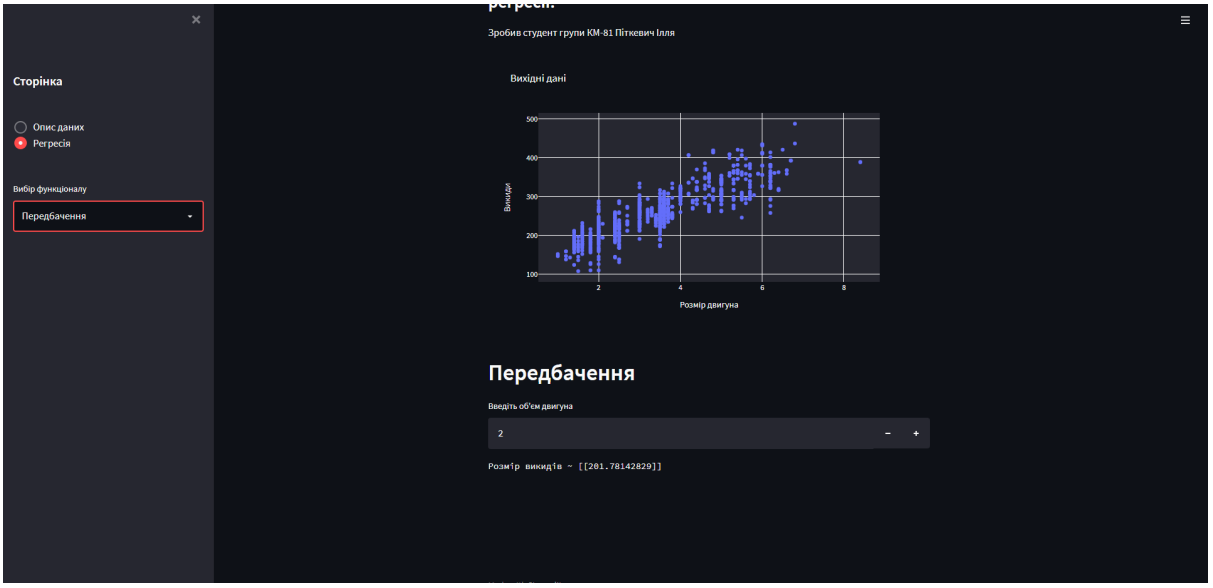
Програма являє собою веб-сайт, що відображається в браузері.



Мається 2 сторінки, перша для опису даних (перевірка даних та базовий розвідувальний аналіз), друга являє собою поліноміальну регресію з різними ступенями та власне функцію прогнозування.







Сама реалізація будується на основі бібліотеки Streamlit. В нас є 4 файли: app.py(головний), multiapp.py(реалізує логіку багатосторінкового сайту, бо strimleat сам по собі не підтримує це), pre_work.py (утилізує опис даних та розвідувальний аналіз), regression.py(власне реалізує передбачення та поліноміальну регресію з двома ступенями - 2 та 3).
Щоб запустити проект потрібно встановити всі залежності з файлу requirements.txt - `pip install -r requirements.txt`
Для запуску виконати - `streamlit run app.py`

Висновки

В результаті ми отримали готову програмну реалізацію поліноміальної регресії для прогнозування викидів вуглецю в залежності від об‘єму двигуна автівки. UI був побудований на streamlit. Більш того, хочеться зазначити, що реалізована модель працює адекватно, тобто прогнозування відбувається передбачувано.

Додаток. Код програми

- app.py

```
import streamlit as st
from multiapp import MultiApp
from pre_work import pre_work
from regression import regression

st.set_page_config(page_title='Polynomial regression', page_icon='🖐')
st.subheader("Прогнозування викиду вуглецю залежно від об'єму двигуна
автомобіля в Канаді на основі поліноміальної регресії.")
st.markdown("Зробив студент групи КМ-81 Піткевич Ілля")

app = MultiApp()
app.add_app("Опис даних", pre_work)
app.add_app("Регресія", regression)

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

- multiapp.py

```
import streamlit as st



class MultiApp:
    def __init__(self):
        self.apps = []

    def add_app(self, title, func):
        """Adds a new application.
        Parameters
        ----------
        func:
            the python function to render this app.
        title:
            title of the app. Appears in the dropdown in the sidebar.
        """
        self.apps.append({
            "title": title,
            "function": func
        })

    def run(self):
```

```
        st.sidebar.header('Сторінка')
        app = st.sidebar.radio(
            '',
            self.apps,
            format_func=lambda app: app['title'])

        app['function']()
```

- pre_work.py

```
import streamlit as st
import plotly.graph_objects as go
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

def pre_work():
    st.sidebar.header('')
    SHOW_DESCRIBE_DATA = st.sidebar.checkbox("Показати опис даних")
    SHOW_EXPLORATORY_DATA = st.sidebar.checkbox("Показати
розвідувальний аналіз")
    fig = go.Figure()
    df = pd.read_csv("FuelConsumption.csv")
    st.dataframe(df.head(50))

    def draw_histograms(dataframe, features, rows, cols):
        fig = plt.figure(figsize=(20, 20))
        for i, feature in enumerate(features):
            ax = fig.add_subplot(rows, cols, i + 1)
            dataframe[feature].hist(bins=20, ax=ax,
facecolor='midnightblue')
            ax.set_title(feature + " Distribution", color='DarkRed')
            ax.set_yscale('log')
        st.pyplot(fig)

    if SHOW_DESCRIBE_DATA:
        st.header("DATA QUALITY CHECK")

        st.caption("Відсоток відсутніх значень у кожному стовпці")
        st.code(round(100 * (df.isnull().sum() / len(df)),
2).sort_values(ascending=False))

        st.caption("Відсоток відсутніх значень у кожному рядку")
```

```
        st.code(round(100 * (df.isnull().sum(axis=1) / len(df)),
2).sort_values(ascending=False))

        st.caption("Опис стовпців")
        st.markdown("""
        - **MODELYEAR** (рік, в якому випущена автівка), наприклад 2013
        - **MAKE** (марка) наприклад Toyota
        - **MODEL** (модель), наприклад HILUX
        - **VEHICLE CLASS** (тип автівки), наприклад кроссовер
        - **ENGINE SIZE** (об'єм двигуна), наприклад 4.2
        - **CYLINDERS** (кількість циліндрів), наприклад 6
        - **TRANSMISSION** (трансміссія), наприклад A6
        - **FUEL CONSUMPTION in CITY**(L/100 km) (споживання топлива в
місті на 100 км)
        - **FUEL CONSUMPTION in HWY** (L/100 km) (споживання топлива на
автомагістралі на 100 км)
        - **FUEL CONSUMPTION COMB** (L/100 km)(споживання топлива
комбіноване на 100 км)
        - **CO2 EMISSIONS** (g/km) (викид вуглецю грам/км)
        """)

    if SHOW_EXPLORATORY_DATA:
        st.header("EXPLORATORY DATA ANALYSIS")

        st.caption("Надає описову статистику, яка узагальнює центральну
тенденцію, дисперсію та форму.")
        st.code(df.describe())

        draw_histograms(df, ["FUELCONSUMPTION_COMB",
"FUELCONSUMPTION_HWY", "FUELCONSUMPTION_CITY",
                             "CO2EMISSIONS", "CYLINDERS"], 8, 4)

        fig = plt.figure()
        sns.heatmap(df.corr(), annot=True)
        st.pyplot(fig)
```

- regression.py

```
import plotly.graph_objects as go
import pandas as pd
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn import linear_model
import streamlit as st
```

```
def regression():
    st.sidebar.header('')
    fig = go.Figure()
    chart_visual = st.sidebar.selectbox('Вибір функціоналу',
                                        ('2-го порядку', '3-го
порядку', 'Передбачення'))
    df = pd.read_csv("FuelConsumption.csv")
    cdf = df[['ENGINESIZE', 'CYLINDERS', 'FUELCONSUMPTION_COMB',
'CO2EMISSIONS']]
    if True:
        fig.add_trace(go.Scatter(x=cdf.ENGINESIZE, y=cdf.CO2EMISSIONS,
mode='markers', ))
        fig.update_layout(title=f'Вихідні дані',
                          xaxis_title='Розмір двигуна',
                          yaxis_title='Викиди')
        st.plotly_chart(fig, use_container_width=True)

        msk = np.random.rand(len(df)) < 0.8
        train = cdf[msk]
        test = cdf[~msk]

        train_x = np.asanyarray(train[['ENGINESIZE']])
        train_y = np.asanyarray(train[['CO2EMISSIONS']])

        test_x = np.asanyarray(test[['ENGINESIZE']])
        test_y = np.asanyarray(test[['CO2EMISSIONS']])

        fig = go.Figure()
        if chart_visual == '2-го порядку':
            poly = PolynomialFeatures(degree=2)
            train_x_poly = poly.fit_transform(train_x)
            clf = linear_model.LinearRegression()
            train_y_ = clf.fit(train_x_poly, train_y)
            # The coefficients
            st.markdown(f'Коефіцієнти: {clf.coef_}')
            st.markdown(f'Перетин: {clf.intercept_}')

            fig.add_trace(go.Scatter(x=cdf.ENGINESIZE,
y=cdf.CO2EMISSIONS, mode='markers', name="Initial Data"))
            XX = np.arange(0.0, 10.0, 0.1)
```

```
            yy = clf.intercept_[0] + clf.coef_[0][1] * XX +
clf.coef_[0][2] * np.power(XX, 2)
            fig.add_trace(go.Scatter(x=XX, y=yy,
                                     mode='lines',
                                     name='Polynomial Regression'))

        elif chart_visual == '3-го порядку':
            poly3 = PolynomialFeatures(degree=3)
            train_x_poly3 = poly3.fit_transform(train_x)
            clf3 = linear_model.LinearRegression()
            train_y3_ = clf3.fit(train_x_poly3, train_y)
            # The coefficients
            st.markdown(f'Коефіцієнти: {clf3.coef_}')
            st.markdown(f'Перетин: {clf3.intercept_}')
            fig.add_trace(go.Scatter(x=cdf.ENGINESIZE,
y=cdf.CO2EMISSIONS, mode='markers', name="Initial Data"))
            XX = np.arange(0.0, 10.0, 0.1)
            yy = clf3.intercept_[0] + clf3.coef_[0][1] * XX +
clf3.coef_[0][2] * np.power(XX, 2) + clf3.coef_[0][
                3] * np.power(
                XX, 3)
            fig.add_trace(go.Scatter(x=XX, y=yy,
                                     mode='lines',
                                     name='Polynomial Regression'))

        elif chart_visual == 'Передбачення':
            poly3_ = PolynomialFeatures(degree=3)
            train_x_poly3 = poly3_.fit_transform(train_x)
            clf3_ = linear_model.LinearRegression()
            train_y3_ = clf3_.fit(train_x_poly3, train_y)
            st.header("Передбачення")
            number = st.number_input("Введіть об'єм двигуна", value=2)
            test_x_poly3 = poly3_.fit_transform([[number]])
            test_y3_ = clf3_.predict(test_x_poly3)
            st.text(f"Розмір викидів ~ {test_y3_}")

        if chart_visual != 'Передбачення':
            fig.update_layout(title=f'Поліномільна регресія
{chart_visual}',
                              xaxis_title='Розмір двигуна',
                              yaxis_title='Викиди')

            st.plotly_chart(fig, use_container_width=True)
```