Project Report On

Economic Growth: A Machine Learning Approach To Gdp Per Capita Prediction

INTRODUCTION

1.1 Overview

Gross Domestic Product is cited as a vital and most widely accepted economic indicator which not only helps in diagnosing the problems related to the economy but also correcting it. The usage of the gross domestic product as a measure of the market price of ultimate services and products that are produced over a selected amount of time will definitely continue to owe an abundance to the producing age. To policy makers and statisticians especially, gross domestic product helps in conveying data about the economy in particular and thereby notifying about a country's economic health. This paper makes an attempt to expedite the process of prediction of Gross Domestic Product.

1.2 Purpose

We are here proposing a system that will be based on Decision Tree, Random Forest and BootStrap algorithms and will forecast the GDP of the Nation. We will use data from Agriculture Production, Manufacturing Industry and Service Industry. These are the sectors of the main mainstream that are responsible for the effect on GDP forecasting. Also we will use Purchasing Power Parity as an input element for better prediction of GDP of a nation. We will use multiple algorithms for increasing and improving accuracy.

2.LITERATURE SURVEY

2.1 Existing Problem

Currently there is no automatic system that will predict GDP of whole country. There experts appointed by Government to study the information regarding multiple sectors of financial income providers. Currently it is critical work to analyze every field of industry running in country. Huge human work power need for this work. Again it doesn't guarantee accuracy of work results and output. GDP which is very most important index for any country in terms of planning of various things. Wrong or miss leading decision can effect on GDP index. It is hard to think in and cover multiple aspects at same time for human.

2.2 Proposed Solution

We are here proposing a system that will be based on Decision Tree, Random Forest and BootStrap algorithms and will forecast the GDP of the Nation. We will use data from Agriculture Production, Manufacturing Industry and Service Industry. These are the sectors of the stream that are responsible for the effect on GDP forecasting. Also we will use Purchasing Power Parity as an input element for better prediction of GDP of a nation. We will use multiple algorithms for increasing and improving accuracy.

Advantages of proposed system

• It would help in predicting the GDP of our country.

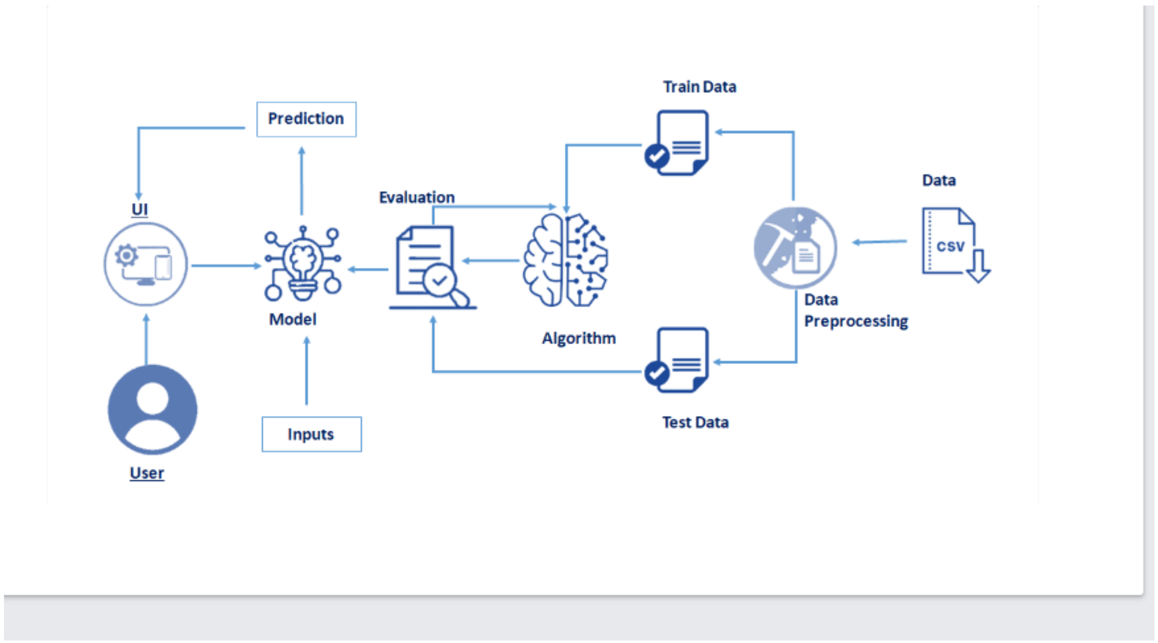
• It would help in making financial decisions and investments decisions with much ease.

• It would help in career guidance indirectly or directly.

• It would be of low cost and highly user friendly.

• It would not require much high maintenance until the parameters change.

3. THEORETICAL ANALYSIS



3.2 Hardware/Software Designing

1.Software Requirements

Downloading of Anaconda Navigator

2.Importing python packages like

a. NumPy Package

b. Pandas

c. seaborn

d. joblib

e. Matplotlib

f. scikit-learn

g. Flask

h. LabelEncoder

I. train_test_split

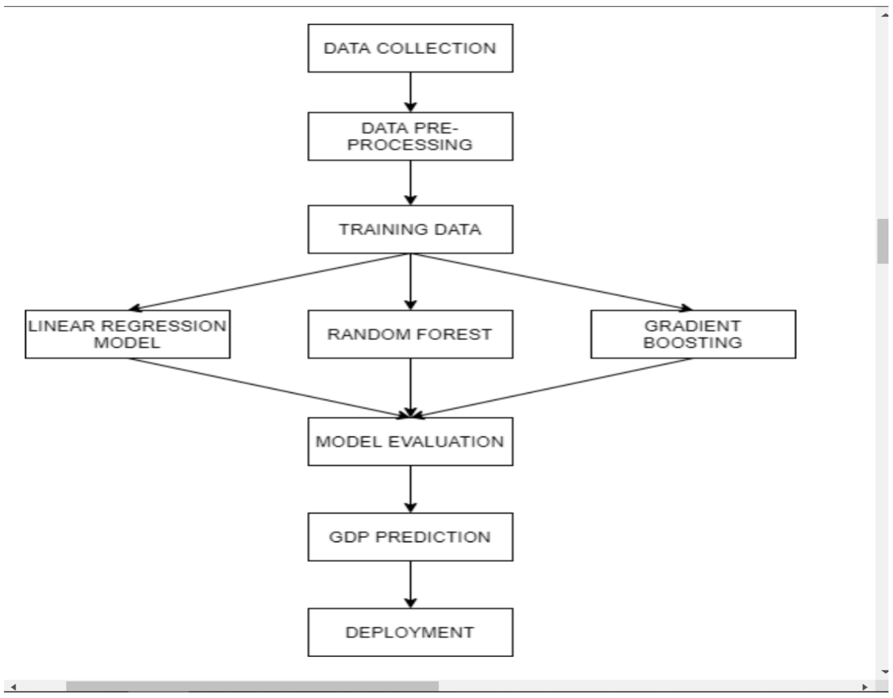
j. LinearRegression

k. RandomForestRegressor

4.EXPERIMENTAL INVESTIGATION

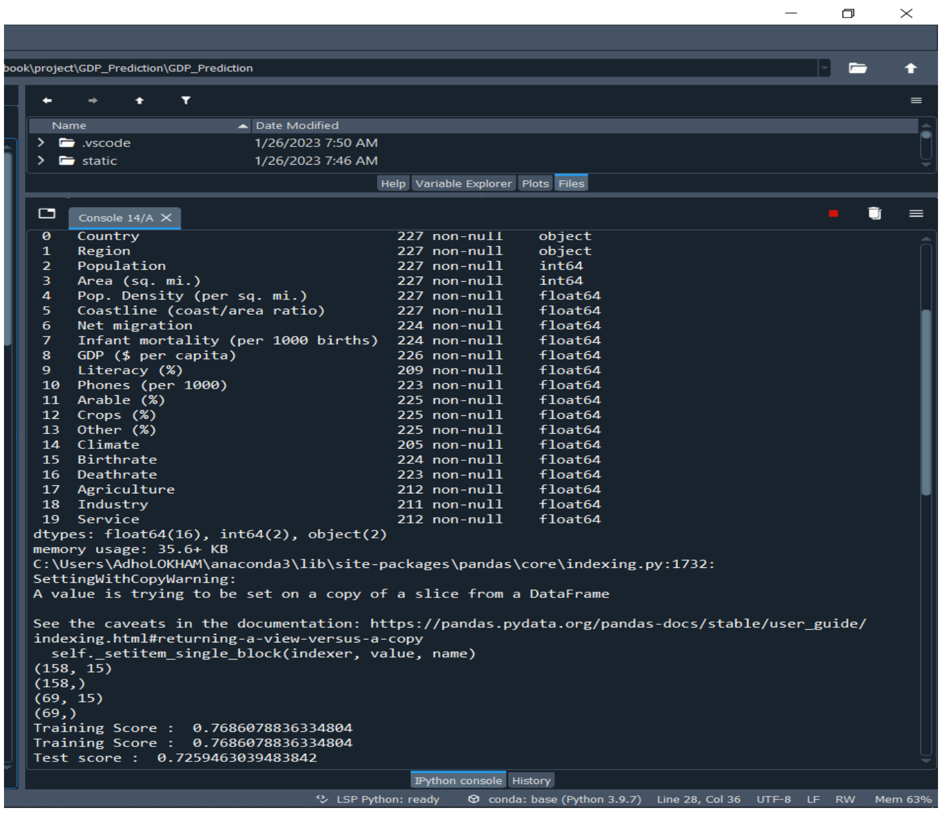
We have imported many libraries. The Python Data Analysis Library (Pandas) is an acronym for "Python Data Analysis Library." Its will be used as the structure for doing reasonable, genuine information investigation in Python. Pandas are built on top of numpy, a package that supports multi-dimensional arrays. Many of the time-consuming, repetitive tasks associated with working with data are made easy with Pandas, including: Data cleaning, Data fill, Data normalization and Joins and merges. Numerical Python (NumPy) is used to solve problems numerically. It also has functions for dealing with matrices and the domain of linear algebra. Benefits of using numpy are fast, fewer loops, cleaner code and improved quality. Seaborn is a matplotlib-based Python data visualization library. It has a high-level interface for producing visually pleasing and insightful statistical graphics. Matplotlib is a Python library that allows you to construct static, animated, and interactive visualizations. With only a few lines of code, you can create publication-quality plots.

5.FLOWCHART



6.RESULT

Fig:1 output of the dataset:



```
book\project\GDP_Prediction\GDP_Prediction
```

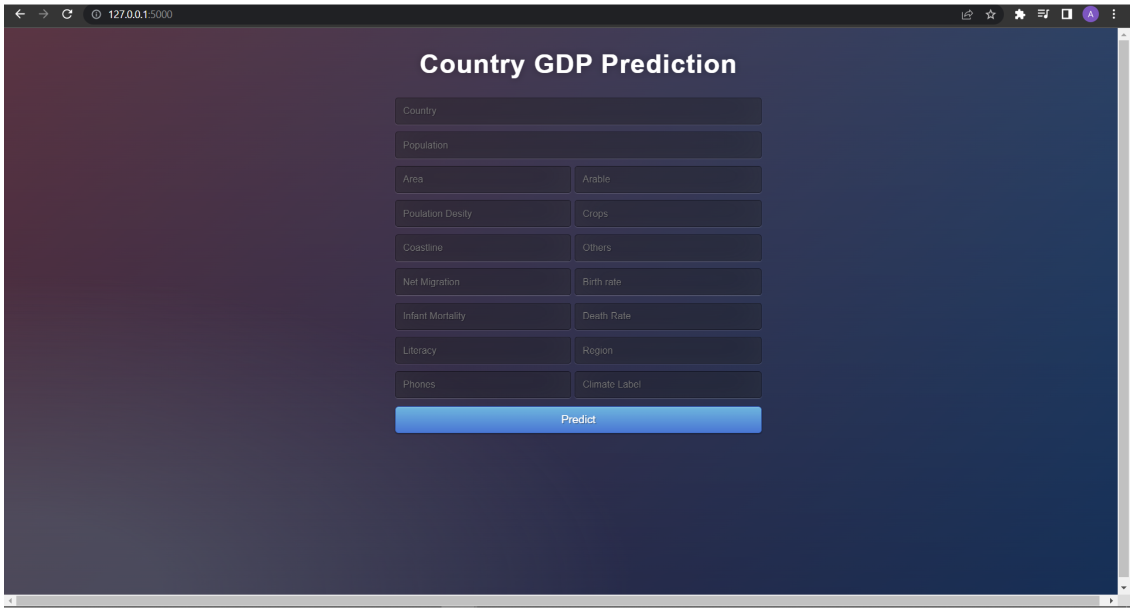
Name	Date Modified
.vscode	1/26/2023 7:50 AM
static	1/26/2023 7:46 AM

```
 0   Country                             227 non-null    object
 1   Region                              227 non-null    object
 2   Population                          227 non-null    int64
 3   Area (sq. mi.)                      227 non-null    int64
 4   Pop. Density (per sq. mi.)          227 non-null    float64
 5   Coastline (coast/area ratio)        227 non-null    float64
 6   Net migration                       224 non-null    float64
 7   Infant mortality (per 1000 births)  224 non-null    float64
 8   GDP ($ per capita)                  226 non-null    float64
 9   Literacy (%)                        209 non-null    float64
 10  Phones (per 1000)                   223 non-null    float64
 11  Arable (%)                          225 non-null    float64
 12  Crops (%)                           225 non-null    float64
 13  Other (%)                           225 non-null    float64
 14  Climate                             205 non-null    float64
 15  Birthrate                           224 non-null    float64
 16  Deathrate                           223 non-null    float64
 17  Agriculture                         212 non-null    float64
 18  Industry                            211 non-null    float64
 19  Service                             212 non-null    float64
dtypes: float64(16), int64(2), object(2)
memory usage: 35.6+ KB
C:\Users\AdhoLOKHAM\anaconda3\lib\site-packages\pandas\core\indexing.py:1732:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/
indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
(158, 15)
(158,)
(69, 15)
(69,)
Training Score :  0.7686078836334804
Training Score :  0.7686078836334804
Test score :  0.7259463039483842
```

```
LSP Python: ready    conda: base (Python 3.9.7)    Line 28, Col 36    UTF-8    LF    RW    Mem 63%
```

Fig 2: Web Application view :

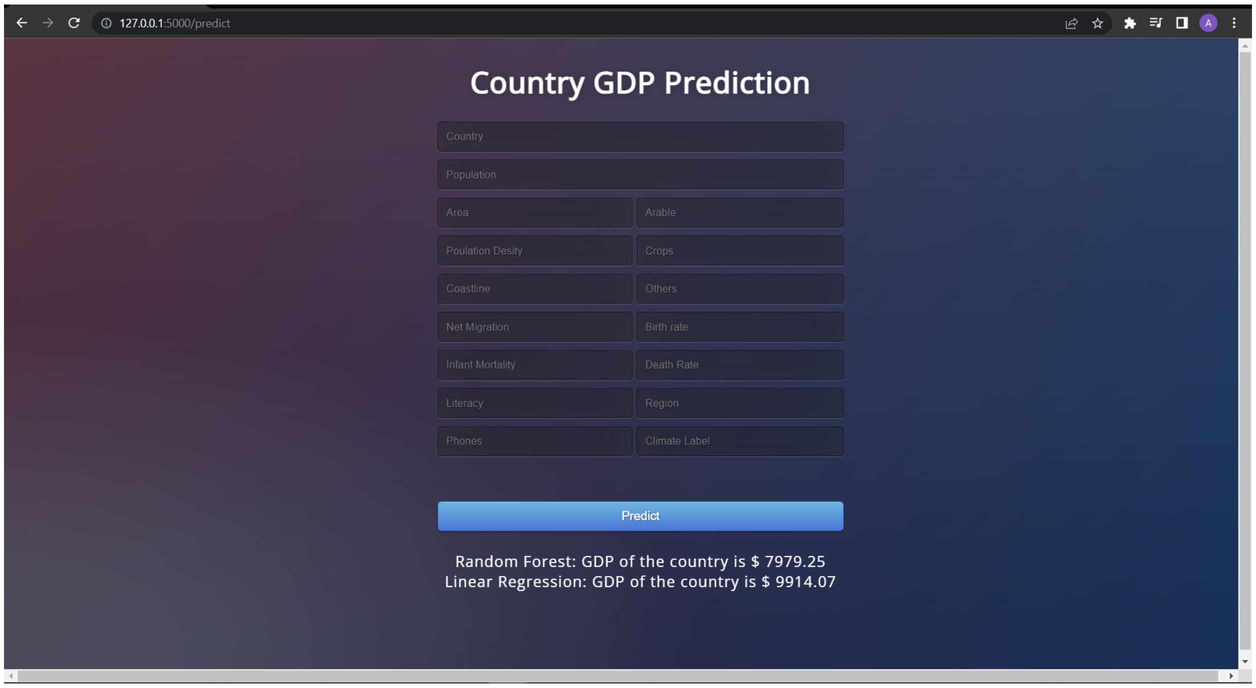


```
127.0.0.1:5000
```

Country GDP Prediction

Country	
Population	
Area	Arable
Poulation Desity	Crops
Coastline	Others
Net Migration	Birth rate
Infant Mortality	Death Rate
Literacy	Region
Phones	Climate Label

Fig 3: Predicting the GDP:



Random Forest: GDP of the country is $ 7979.25
Linear Regression: GDP of the country is $ 9914.07

7.ADVANTAGES AND DISADVANTAGES

Advantages

- It reduces overfitting in decision trees and helps to improve the accuracy
- It is flexible to both classification and regression problems
- It works well with both categorical and continuous values
- It automates missing values present in the data
- Normalizing of data is not required as it uses a rule-based approach.

Disadvantages

- It requires much computational power as well as resources as it builds numerous trees to combine their outputs.
- It also requires much time for training as it combines a lot of decision trees to determine the class.
- Due to the ensemble of decision trees, it also suffers interpretability and fails to determine the significance of each variable.

8.APPLICATIONS

The accuracy obtained by linear regression algorithm is 82% and by random forest is 87%. On the basis of the optimization process, the machine learning algorithm “Gradient Boosting” utilized during this project worked well with the accuracy 89% in order to predict the true GDP per capita.

9.CONCLUSION

We explored all the supervised regression models in order to get the best fitting models. We have trained the model using Linear Regression, Random Forest and Gradient Boosting machine learning algorithms and also estimated the performance of these models. Evaluation is done using MAE and RMSE techniques and then compared all three models to get a clear overview of performance. The accuracy obtained by linear regression algorithm is 82% and by random forest is 87%. On the basis of the optimization process, the machine learning algorithm “Gradient Boosting” utilized during this project worked well with the accuracy 89% in order to predict the true GDP per capita. Finally deployed the highest accuracy model to “GDP Estimation Tool” which estimates and forecasts GDP of a country just by giving some attribute as input for that country.

10. FUTURE SCOPE

In the future, we can model the system using a vector auto regression algorithm which predicts the future GDP per capita based upon the GDP per capita in the preceding years.

Here we tried to make research on and implement almost every most possible aspect that will make impact on our proof of concept. In future work we can think of more other sectors that are making effect on GDP. After deep and more working on algorithms we can implement new algorithms that will work strongly with data.

11.BIBLIOGRAPHY

1.https://www.kaggle.com

2. https://github.com

APPENDIX:

Source code:

```
import numpy as np

from flask import Flask, request, jsonify, render_template

import pickle

import sklearn.externals as extjoblib

import joblib

app = Flask(__name__)

model = joblib.load('Random.pkl')

model1 = joblib.load('Linear.pkl')

@app.route('/')

def home():

    return render_template('index.html')


@app.route('/predict',methods=['POST'])

def predict():

    '''

    For rendering results on HTML GUI

    '''


    int_features = [float(x) for x in request.form.values()]

    final_features = [np.array(int_features)]


    prediction = model.predict(final_features)

    prediction1 = model1.predict(final_features)
    output = round(prediction[0], 2)
    output1 = round(prediction1[0], 2)
    return render_template('index.html', prediction_text='Random Forest: GDP of the country is  $
{}'.format(output), prediction_text2='Linear Regression: GDP of the country is  $
{}'.format(output1))
@app.route('/country',methods=['POST'])
def country():
    '''
    For rendering results on HTML GUI
    '''

    text=request.form.get("country")

    return render_template('index.html', prediction_text3=text)
@app.route('/predict_api',methods=['POST'])
def predict_api():
    '''
    For direct API calls trought request
    '''
    data = request.get_json(force=True)
    prediction = model.predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)
```

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

```python
# -*- coding: utf-8 -*-
"""GDP.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1VKGQ2gzGRmH0VyIv-N-QSs7cDFGfAdsA
"""

import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
import pickle
```

```
world=pd.read_csv("countries of the world.csv",decimal=',')


world.info()

world.head()


world.value_counts('Region')


#world.value_counts('Region_label')


world.describe()


world.isnull().sum()


for col in world.columns.values:
    if world[col].isnull().sum() == 0:
        continue
    if col == 'Climate':
        guess_values = world.groupby('Region')['Climate'].apply(lambda x: x.mode().max())
    else:
        guess_values = world.groupby('Region')[col].median()
    for region in world['Region'].unique():
        world[col].loc[(world[col].isnull())&(world['Region']==region)] = guess_values[region]


LE = LabelEncoder()
```

```
world['Region_label'] = LE.fit_transform(world['Region'])

world['Climate_label'] = LE.fit_transform(world['Climate'])

world.head()


world['Region_label'].unique()


world['Region_label']


train, test = train_test_split(world, test_size=0.3, shuffle=True)

training_features = ['Population', 'Area (sq. mi.)',

       'Pop. Density (per sq. mi.)', 'Coastline (coast/area ratio)',

       'Net migration', 'Infant mortality (per 1000 births)',

       'Literacy (%)', 'Phones (per 1000)',

       'Arable (%)', 'Crops (%)', 'Other (%)', 'Birthrate',

       'Deathrate','Region_label' ,

       'Climate_label']

target = 'GDP ($ per capita)'

train_X = train[training_features]

train_Y = train[target]

test_X = test[training_features]

test_Y = test[target]


train, test = train_test_split(world, test_size=0.3, shuffle=True)

training_features = ['Population', 'Area (sq. mi.)',
      'Pop. Density (per sq. mi.)', 'Coastline (coast/area ratio)',
      'Net migration', 'Infant mortality (per 1000 births)',
      'Literacy (%)', 'Phones (per 1000)',
      'Arable (%)', 'Crops (%)', 'Other (%)', 'Birthrate',
      'Deathrate', 'Region_label',
      'Climate_label']
target = 'GDP ($ per capita)'
train_X = train[training_features]
train_Y = train[target]
test_X = test[training_features]
test_Y = test[target]

print(train_X.shape)
print(train_Y.shape)
print(test_X.shape)
print(test_Y.shape)

model1 = LinearRegression()
model1.fit(train_X, train_Y)
train_pred_Y = model1.predict(train_X)
test_pred_Y = model1.predict(test_X)

print('Training Score : ',model1.score(train_X,train_Y))
#print(f'Test score : ',r2_score(test_pred_Y,test_Y))
```

```
model = RandomForestRegressor(n_estimators = 100,
                              max_depth = 6,
                              min_weight_fraction_leaf = 0.05,
                              max_features = 0.8,
                              random_state = 42)
model.fit(train_X, train_Y)
train_pred_Y = model.predict(train_X)
test_pred_Y = model.predict(test_X)

from sklearn.metrics import r2_score

print('Training Score : ',model1.score(train_X,train_Y))
print(f'Test score : ',r2_score(test_pred_Y,test_Y))

df = pd.DataFrame(columns = training_features)

df1=[[31056997.0,647500.0,48.0,0,23.06,163.07,36.0,3.2,12.13,0.22,87.65,46.6,20.34,0,0]]
model.predict(df1)

df=[[3581655,28748, 124.6,  1.26    ,4.93   ,21.52,86.5 ,71.2   ,21.09  ,4.42   ,74.49
,15.11,5.22,3,4]]
model.predict(df)
```

```
dfk=[[500000000.0,  3287263.0   ,152.0, 2.00,   0.00,   5.00,   99.00,  1000.00,    60.0,   10.0,
30.00,  10.00,  5.00,   0.0,    0.0,]]

model.predict(df)


from joblib import Parallel, delayed

import joblib



# Save the model as a pickle in a file

joblib.dump(model1, 'Linear.pkl')


# Load the model from the file

knn_from_joblib = joblib.load('Linear.pkl')


# Use the loaded model to make predictions

knn_from_joblib.predict(dfk)
```