Project Stage - IV (Basic Machine Learning) ddl: 04/28/2023

Goals

The goal of Stage IV is to utlize machine learning and statistical models to predict the trend of COVID-19 cases / deaths.

Tasks for Stage IV:

Task 1: (70 pts)

- Team: (30)
  - Develop Linear and Non-Linear (polynomial) regression models for predicting cases and deaths in US.
    - Start your data from the first day of infections in US. X-Axis - number of days since the first case, Y-Axis - number of new cases and deaths.
    - Calculate and report Root Mean Square Error (RMSE) for your models (linear and non-linear). Discuss bias versus variance tradeoff.
    - Plot trend line along for the data along with the forecast of 1 week ahead.
    - Describe the trends as compared to other countries.
- Member: (40 pts)
  - Utilize Linear and Non-Linear (polynomial) regression models to compare trends for a single state (each member should choose different state) and its counties (top 5 with highest number of cases). Start your data from the first day of infections.
    - X-Axis - number of days since the first case, Y - Axis number of new cases and deaths. Calcluate error using RMSE.
    - Identify which counties are most at risk. Model for top 5 counties with cases within a state and describe their trends.
    - Utilize the hospital data to calculate the point of no return for a state. Use percentage occupancy / utilization to see which states are close and what their trend looks like.
    - Perform hypothesis tests on questions identified in Stage II
      - e.x. *Does higher employment data (overall employment numbers) lead to higher covid case numbers or more rapid increase in covid cases.*. Here you would compare the covid cases to the state or county level enrichment data to prove or disprove your null hypothesis. In this case there will be a two tail - two sample t-test to see if there is a difference and then one-tail - two sample t-test to show higher or lower.
    - Depending on your type of data you can also perform Chi-square test for categorical hypothesis testing.

Task 2: (30 pts)

- Member:
    - For each of the aforemention analysis plot graphs,
        - trend line
        - confidence intervals (error in prediction)
        - prediction path (forecast)

Deliverable

- Each member creates separate notebooks for member tasks. Upload all notebooks and reports to Canvas. Do not submit to Github, at least before the submission deadline, to avoid potential plagiarism.

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from math import sqrt
from sklearn.svm import SVR
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
confirmed_cases = pd.read_csv('data/covid_confirmed_usafacts.csv')
confirmed_cases.drop(confirmed_cases[confirmed_cases["County Name"].str.contains("S
confirmed_cases = confirmed_cases[(confirmed_cases.iloc[:, 4:] >= 0).all(1)]
confirmed_cases
```

Out[ ]:

	countyFIPS	County Name	State	StateFIPS	2020-01-22	2020-01-23	2020-01-24	2020-01-25	2020-01-26	2020-01-27	...	20 01
1	1001	Autauga County	AL	1	0	0	0	0	0	0	...	19
2	1003	Baldwin County	AL	1	0	0	0	0	0	0	...	68
3	1005	Barbour County	AL	1	0	0	0	0	0	0	...	7
4	1007	Bibb County	AL	1	0	0	0	0	0	0	...	7
5	1009	Blount County	AL	1	0	0	0	0	0	0	...	17
...	...	...	...	...	...	...	...	...	...	...	...	
3188	56037	Sweetwater County	WY	56	0	0	0	0	0	0	...	12
3189	56039	Teton County	WY	56	0	0	0	0	0	0	...	11
3190	56041	Uinta County	WY	56	0	0	0	0	0	0	...	6
3191	56043	Washakie County	WY	56	0	0	0	0	0	0	...	2
3192	56045	Weston County	WY	56	0	0	0	0	0	0	...	1

3142 rows × 1095 columns

In [ ]:

```
confirmed_deaths = pd.read_csv('data/covid_deaths_usafacts.csv')
confirmed_deaths.drop(confirmed_deaths[confirmed_deaths["County Name"].str.contains
confirmed_deaths = confirmed_deaths[(confirmed_deaths.iloc[:, 4:] >= 0).all(1)]
confirmed_deaths
```

Out[ ]:

	countyFIPS	County Name	State	StateFIPS	2020-01-22	2020-01-23	2020-01-24	2020-01-25	2020-01-26	2020-01-27	...	20 01
1	1001	Autauga County	AL	1	0	0	0	0	0	0	...	
2	1003	Baldwin County	AL	1	0	0	0	0	0	0	...	
3	1005	Barbour County	AL	1	0	0	0	0	0	0	...	
4	1007	Bibb County	AL	1	0	0	0	0	0	0	...	
5	1009	Blount County	AL	1	0	0	0	0	0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
3188	56037	Sweetwater County	WY	56	0	0	0	0	0	0	...	
3189	56039	Teton County	WY	56	0	0	0	0	0	0	...	
3190	56041	Uinta County	WY	56	0	0	0	0	0	0	...	
3191	56043	Washakie County	WY	56	0	0	0	0	0	0	...	
3192	56045	Weston County	WY	56	0	0	0	0	0	0	...	

3142 rows × 1095 columns

In [ ]:

```
covid_data = pd.merge(confirmed_cases, confirmed_deaths, on=["countyFIPS", "County
#covid_data = pd.merge(confirmed_cases, confirmed_deaths, on=["Type"], suffixes=["_

display(covid_data)
```

	countyFIPS	County Name	State	StateFIPS	2020-01-22_confirmed	2020-01-23_confirmed	2020-01-24_confirmed	2 25_co
0	1001	Autauga County	AL	1	0	0	0	
1	1003	Baldwin County	AL	1	0	0	0	
2	1005	Barbour County	AL	1	0	0	0	
3	1007	Bibb County	AL	1	0	0	0	
4	1009	Blount County	AL	1	0	0	0	
...	...	...	...	...	...	...	...	
3137	56037	Sweetwater County	WY	56	0	0	0	
3138	56039	Teton County	WY	56	0	0	0	
3139	56041	Uinta County	WY	56	0	0	0	
3140	56043	Washakie County	WY	56	0	0	0	
3141	56045	Weston County	WY	56	0	0	0	

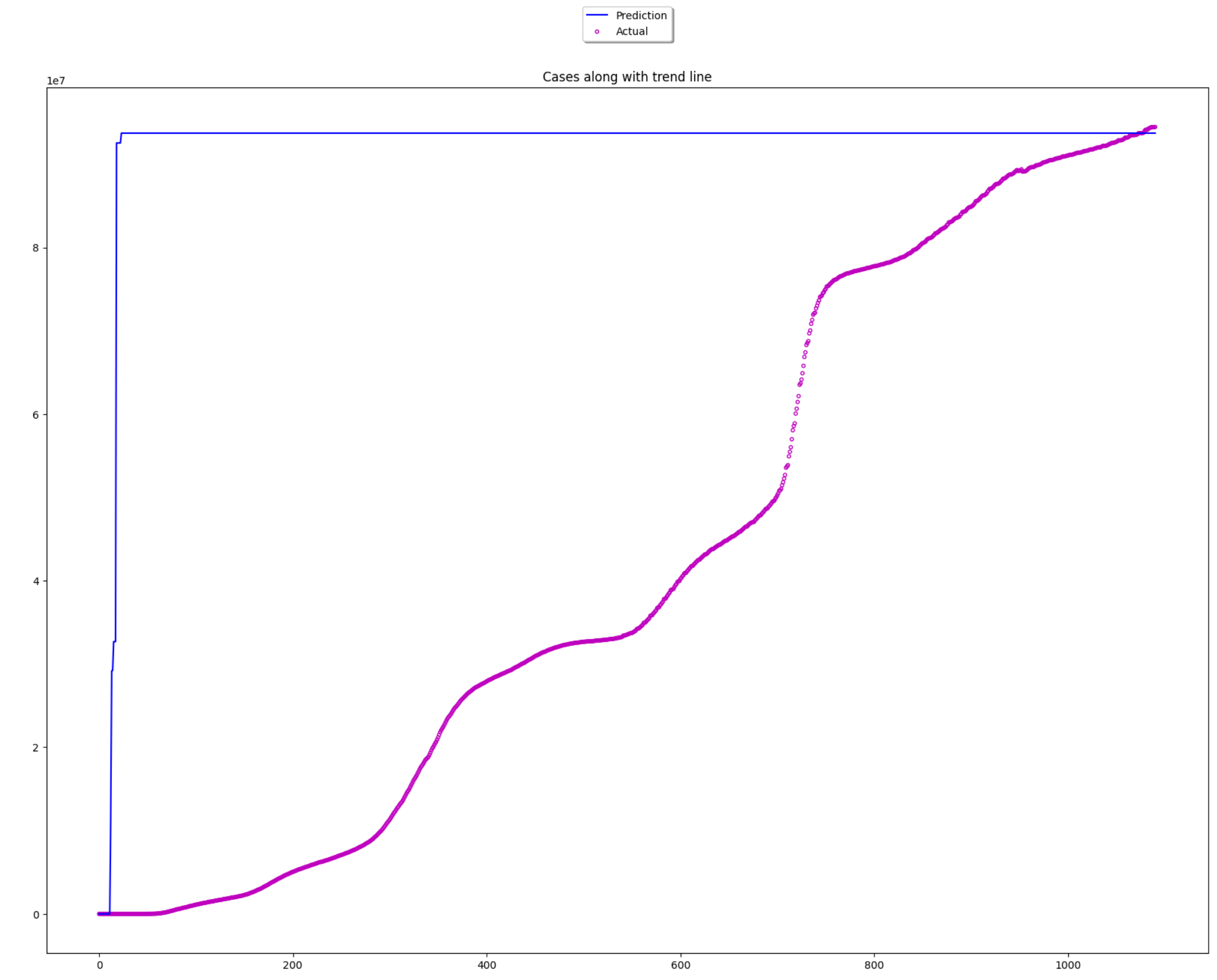
3142 rows × 2186 columns

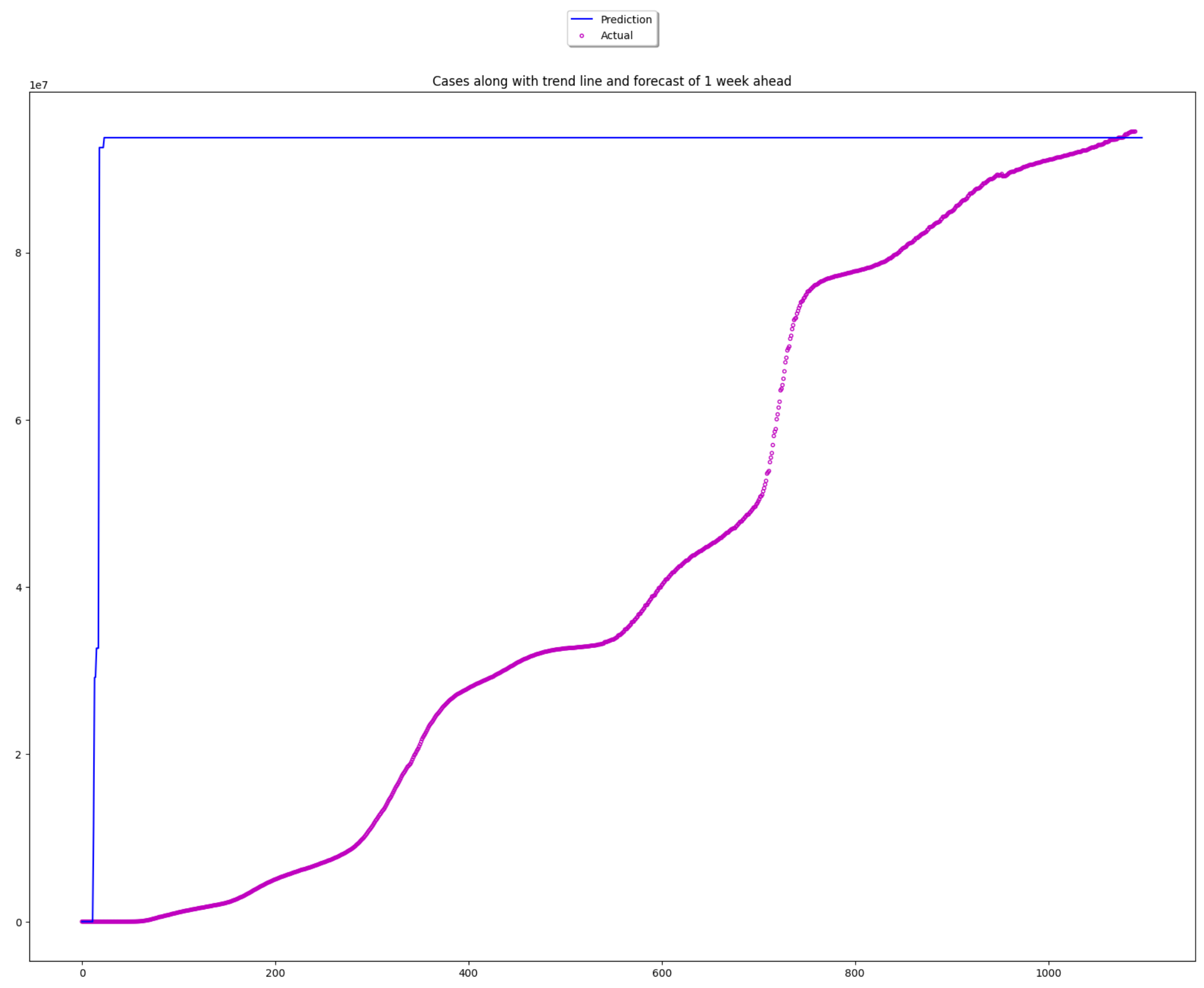
```
In [ ]: def question1(confirmed_cases):
            group_cases_USA = confirmed_cases.sum()
            group_cases_USA = group_cases_USA.iloc[4:]

            temp = group_cases_USA.index

            X =[]
            for i in range(len(temp)):
                X.append(i)
            Y= group_cases_USA.values
            X = np.array(X)
            X= X.reshape(-1,1)
            Y = Y.astype('int')

            X_train_cases, X_test_cases, y_train_cases, y_test_cases = train_test_split(X,

            lrg = LogisticRegression()
            lrl = LinearRegression()
            lrg.fit(X_train_cases,y_train_cases)
            lrl.fit(X_train_cases,y_train_cases)
```

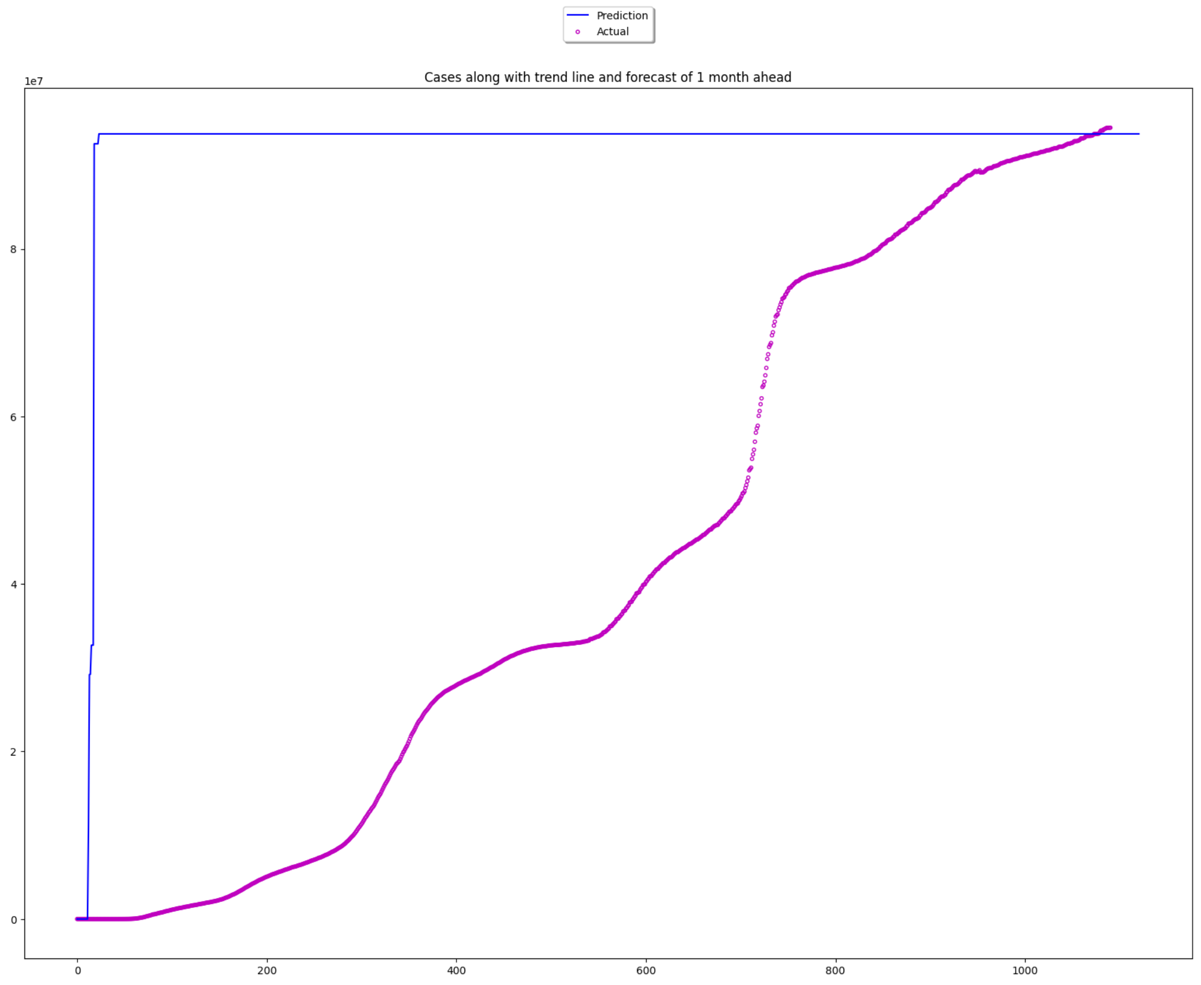
```
    y_pred_lrl = lrl.predict(X_test_cases)
    y_pred_lrg = lrg.predict(X_test_cases)

    rmse_lrl = sqrt( mean_squared_error(y_test_cases, y_pred_lrl))
    rmse_lrg = sqrt(mean_squared_error(y_test_cases, y_pred_lrg))

    print(f'Logistic regression Root Mean Square Error (RMSE): {round(rmse_lrg,2)}'
    print(f'Linear regression Root Mean Square Error (RMSE): {round(rmse_lrl,2)}')

    diff_pred_act = y_test_cases - y_pred_lrg
    sum_diff = sum(diff_pred_act)
    bias = sum_diff/len(y_test_cases)
    print(f'Bias logistic regression model: {round(bias,2)}')

    pred_var = y_pred_lrg.var()
    print(f'Variance logistic regression model: {round(pred_var)}')

    plt.figure(figsize=(20,15))
    plt.title('Cases along with trend line')
    plt.plot(X,
        lrg.predict(X),
        color='b',
        label = 'Prediction')

    plt.scatter(X,Y,facecolor="none",
        edgecolor='m',
        s=10,
           label = 'Actual')
    plt.legend(
    loc="upper center",
    bbox_to_anchor=(0.5, 1.1),
    ncol=1,
    fancybox=True,
    shadow=True,
    )
    plt.show()

    next_week = []
    for i in range(7):
        next_week.append(i + len(X))
    next_week = np.array(next_week)
    next_week= next_week.reshape(-1,1)

    y_next_week = lrg.predict(next_week)
    X_next = np.append(X, next_week)
    X_next = X_next.reshape(-1,1)

    plt.figure(figsize=(20,15))
    plt.title('Cases along with trend line and forecast of 1 week ahead')
    plt.plot(X_next,
            lrg.predict(X_next),
            color='b',
            label = 'Prediction')

    plt.scatter(X,Y,facecolor="none",
            edgecolor='m',
            s=10,
               label = 'Actual')
    plt.legend(
        loc="upper center",
        bbox_to_anchor=(0.5, 1.1),
        ncol=1,
        fancybox=True,
        shadow=True,
    )
    plt.show()


    next_month = []
    for i in range(30):
        next_month.append(i + len(X))
    next_month = np.array(next_month)
    next_month= next_month.reshape(-1,1)

    y_next_month = lrg.predict(next_month)
    X_next = np.append(X, next_month)
    X_next = X_next.reshape(-1,1)

    plt.figure(figsize=(20,15))
    plt.title('Cases along with trend line and forecast of 1 month ahead')
    plt.plot(X_next,
            lrg.predict(X_next),
            color='b',
            label = 'Prediction')

    plt.scatter(X,Y,facecolor="none",
            edgecolor='m',
            s=10,
               label = 'Actual')
    plt.legend(
        loc="upper center",
        bbox_to_anchor=(0.5, 1.1),
        ncol=1,
        fancybox=True,
        shadow=True,
    )
    plt.show()
```

In [ ]:
```
question1(confirmed_cases)
```

```
Logistic regression Root Mean Square Error (RMSE): 9592.16
Linear regression Root Mean Square Error (RMSE): 6198163.37
Bias logistic regression model: -8452189.26
Variance logistic regression model: 156770044027619
```


Prediction
Actual
Cases along with trend line
1e7
0
2
4
6
8
0
200
400
600
800
1000


Prediction
Actual
Cases along with trend line and forecast of 1 week ahead
1e7
8
6
4
2
0
0
200
400
600
800
1000



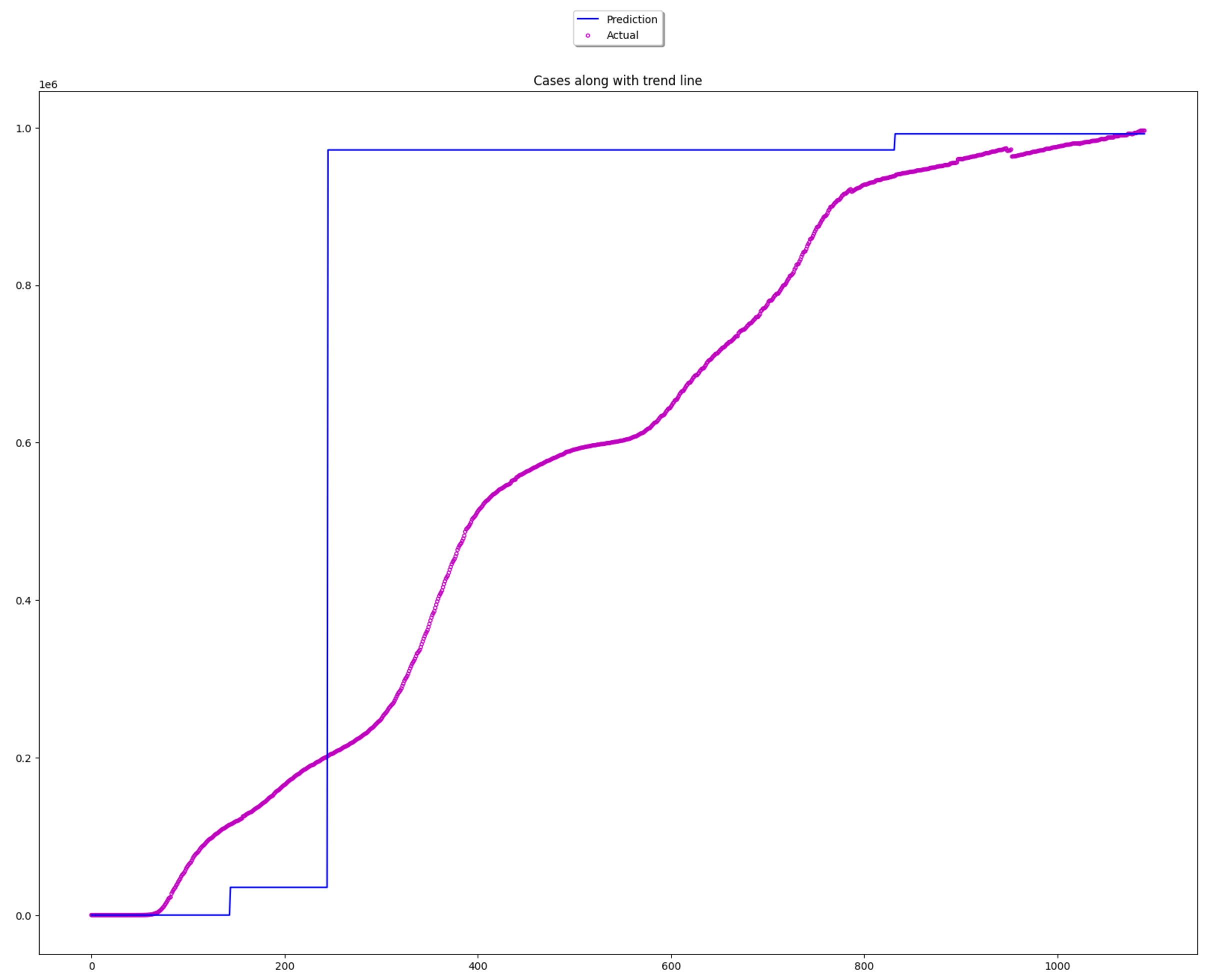
RMSE of both logistic regression and linear regression model decreases as the size of the training set increases, and it decreases the most for the logistic regression.

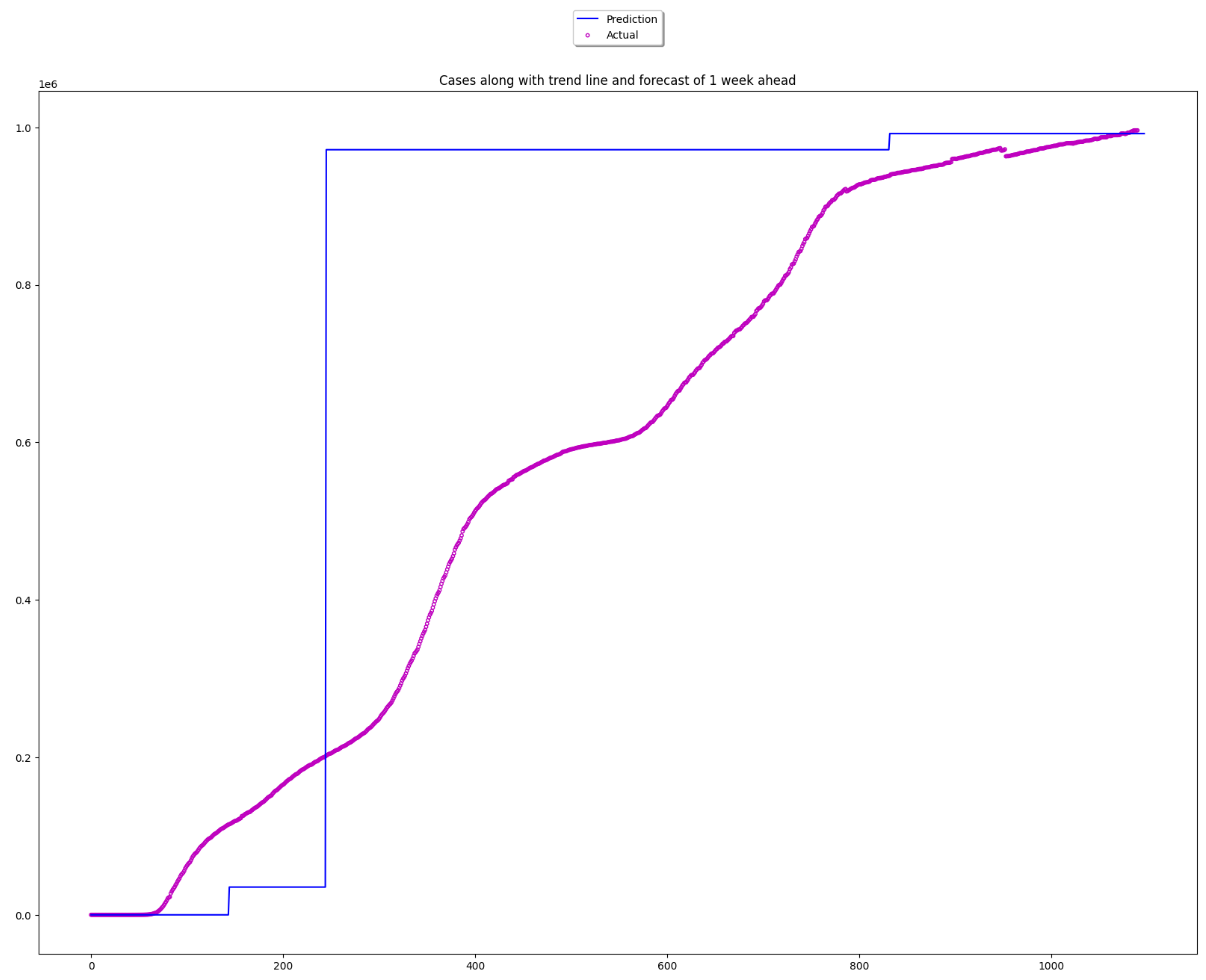
We can see that the calculated bias is very low, in comparison to the high variance. This means that our logistic regression model is underfitting and is not able to fully capture the pattern in our dataset.

Predicting Deaths in USA

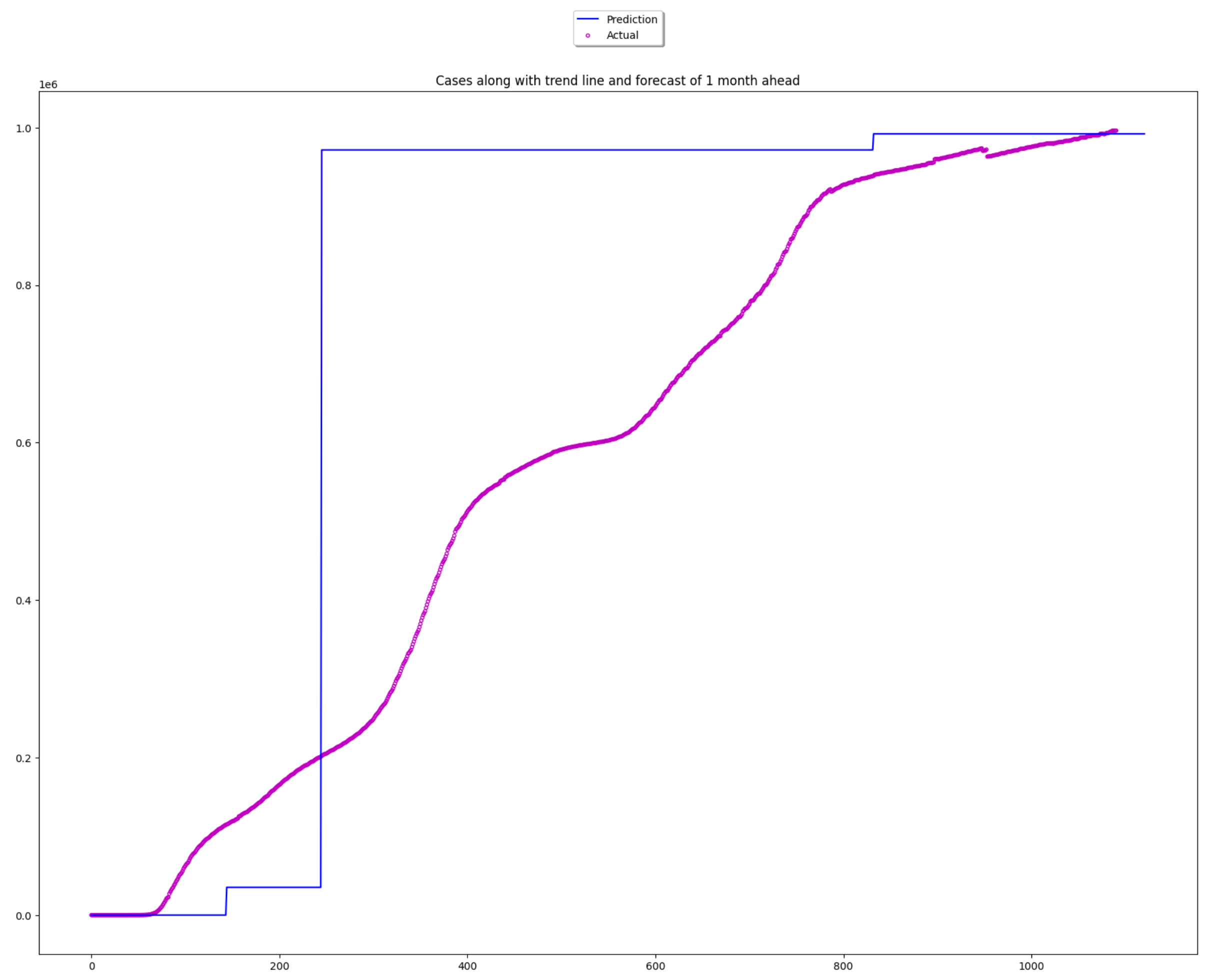
In [ ]:
```
question1(confirmed_deaths)
```

```
Logistic regression Root Mean Square Error (RMSE): 9673.42
Linear regression Root Mean Square Error (RMSE): 62146.88
Bias logistic regression model: -216732.14
Variance logistic regression model: 162724045016
```


Prediction
Actual
Cases along with trend line
1e6
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000



Prediction
Actual
Cases along with trend line and forecast of 1 week ahead
1e6
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000



Comparing the U.S. to other Countries

In [ ]:
```
world_df = pd.read_csv("data/owid-covid-data-cases.csv")
world_df
```

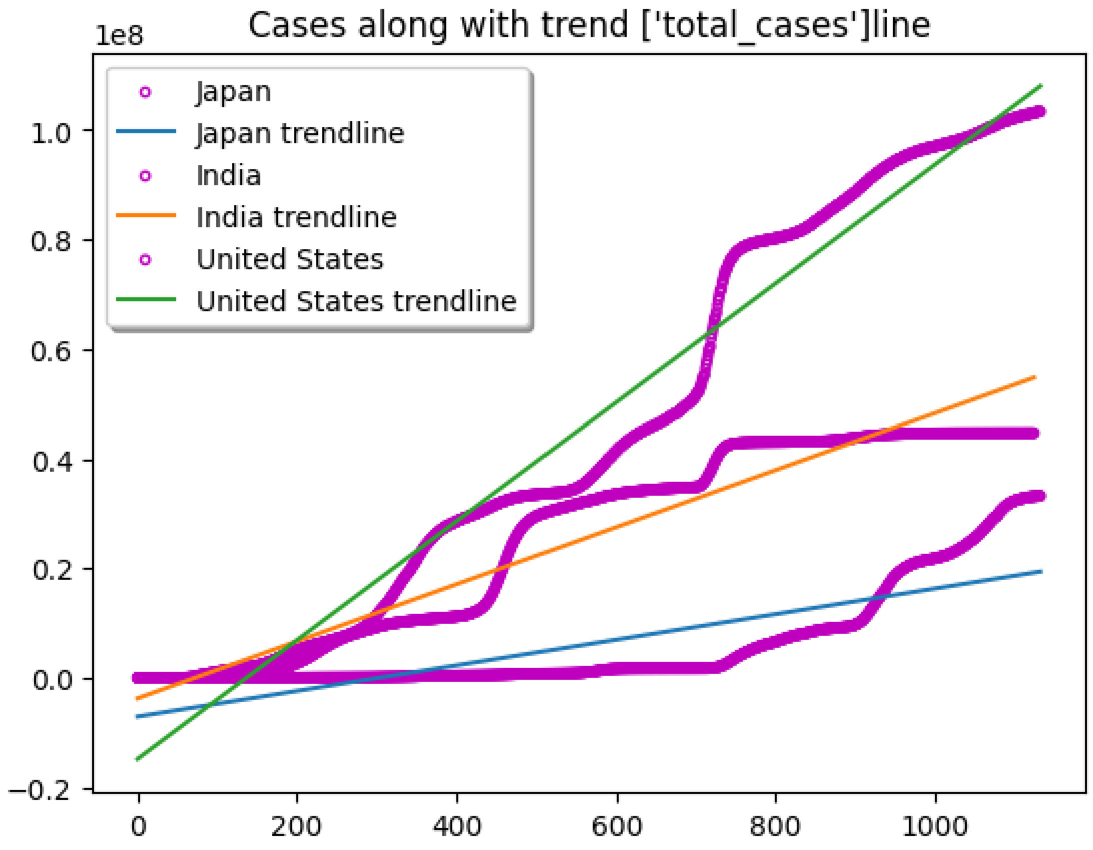
Out[ ]:

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	tota
0	AFG	Asia	Afghanistan	2020-02-24	5.0	5.0	NaN	
1	AFG	Asia	Afghanistan	2020-02-25	5.0	0.0	NaN	
2	AFG	Asia	Afghanistan	2020-02-26	5.0	0.0	NaN	
3	AFG	Asia	Afghanistan	2020-02-27	5.0	0.0	NaN	
4	AFG	Asia	Afghanistan	2020-02-28	5.0	0.0	NaN	
...	...	...	...	...	...	...	...	
260562	ZWE	Africa	Zimbabwe	2023-02-23	263921.0	NaN	NaN	
260563	ZWE	Africa	Zimbabwe	2023-02-24	263921.0	NaN	NaN	
260564	ZWE	Africa	Zimbabwe	2023-02-25	263921.0	NaN	NaN	
260565	ZWE	Africa	Zimbabwe	2023-02-26	263921.0	NaN	NaN	
260566	ZWE	Africa	Zimbabwe	2023-02-27	263921.0	NaN	NaN	

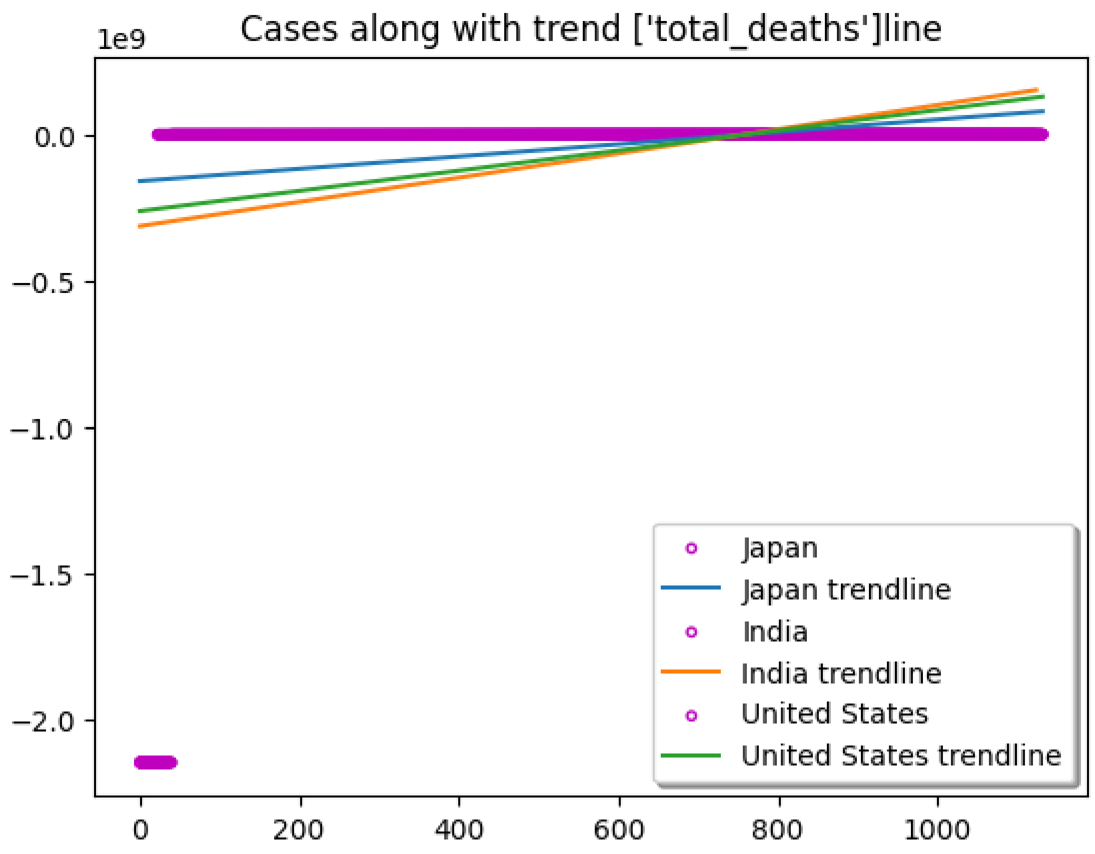
260567 rows × 67 columns

In [ ]:
```python
def question1_world(dataframe, columns, countries):
    for country in countries:
        country_df = dataframe[dataframe['location'] == country][columns]
        temp = country_df.index

        X =[]
        for i in range(len(temp)):
            X.append(i)
        Y = country_df.values
        X = np.array(X)
        X= X.reshape(-1,1)
        Y = Y.astype('int')

        X_train_cases, X_test_cases, y_train_cases, y_test_cases = train_test_split

        lrl = LinearRegression()
        lrl.fit(X_train_cases,y_train_cases)

        y_pred_lrl = lrl.predict(X_test_cases)

        rmse_lrl = sqrt( mean_squared_error(y_test_cases, y_pred_lrl))
        print(f'Linear regression Root Mean Square Error (RMSE) for {country}: {rou

        # Plot the data for the current country
        plt.scatter(X,Y,facecolor="none",
            edgecolor='m',
            s=10,
            label=country)

        plt.plot(X, lrl.predict(X), label=f'{country} trendline')

    plt.title(f'Cases along with trend {columns }line')
    plt.legend(fancybox=True, shadow=True)
    plt.show()

question1_world(world_df, ['total_cases'], ['Japan','India', 'United States'])
question1_world(world_df, ['total_deaths'], ['Japan','India', 'United States'])
```

```
Linear regression Root Mean Square Error (RMSE) for Japan: 6063929.17
Linear regression Root Mean Square Error (RMSE) for India: 4719928.41
Linear regression Root Mean Square Error (RMSE) for United States: 5959016.42
```



```
Linear regression Root Mean Square Error (RMSE) for Japan: 332439305.12
Linear regression Root Mean Square Error (RMSE) for India: 334408215.09
Linear regression Root Mean Square Error (RMSE) for United States: 448910582.28
```



Analysis

Looking at the data plotted above, it goes to show that the United States on average had the highest amount of cases of the past few years. However, compared to contries such as Pakistan and China, they had lower amounts. Which is interesting to note, since China in particular has a higher/denser population then the United States. We believe that this is the result of erroneous reporting from them. The Total deaths, had India place irst, with the United States closely behind. Below is the total amount of deaths based each country.

In [ ]:
```
def analysis_world(dataframe, countries):
    for country in countries:
        country_df = dataframe[dataframe['location'] == country]
        total_cases = country_df['total_cases'].iloc[-1]
        total_deaths = country_df['total_deaths'].iloc[-1]
        population = country_df['population'].iloc[-1]

        print(f"Country: {country}")
        print(f"Total Cases: {total_cases}")
        print(f"Total Deaths: {total_deaths}")
        print(f"Population: {population}")
        print("\n")
analysis_world(world_df, ['China','Japan','India', 'United States'])
```

```
Country: China
Total Cases: 2023904.0
Total Deaths: 87468.0
Population: 1425887360.0


Country: Japan
Total Cases: 33212438.0
Total Deaths: 72328.0
Population: 123951696.0


Country: India
Total Cases: 44687597.0
Total Deaths: 530771.0
Population: 1417173120.0


Country: United States
Total Cases: 103389954.0
Total Deaths: 1119550.0
Population: 338289856.0
```