Hospital Resource Prediction

Predicting hospital bed usage and its effect on COVID-19 fatality rates.

Mahsa Aghaeeaval

Problem Definiton and Our Goal

Implementing various algorithms to create a prediction model that will predict when each country will run out of hospital beds.

Having such information can guide healthcare professionals with managing and allocation of their resources.

We then want to evaluate the effect of low hospital bed availability on fatality rate.

EDA Summary-

Variables from raw data

- What useful variables can we use from our raw data?
  - Number of new cases per day
  - Number of deaths per day
  - Number of recoveries per day
  - Number of tests done per day
  - Mask / Stay at home / school closure / travel restriction mandates
  - Date
  - Hospital capacity
  - Beds used by non-COVID cases
  - Beds used by COVID cases

EDA Summary-

Variables created from raw data

Independent Attributes

- Number of new cases per day
- Number of deaths per day
- Number of recoveries per day

→

- Active cases per day
  - Per population
  - normalized

- Number of tests done per day

→

- Tests per day
  - Per population
  - normalized

- Date
- Number of new cases per day

→

- Days since first case [1,365]

- Mask / Stay at home / school closure / travel restriction mandates

→

- Mandate or no mandate [0,1]

Target Attribute

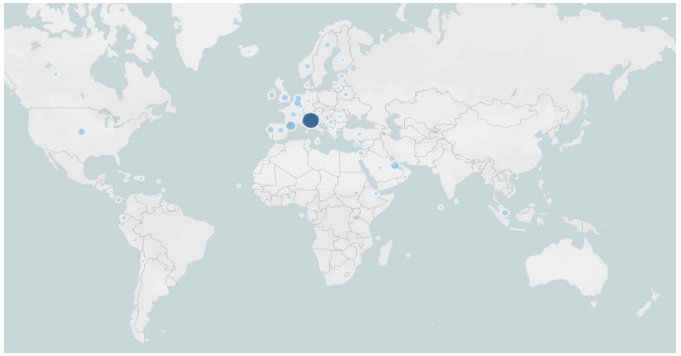
- Hospital capacity
- Beds used by non-COVID cases
- Beds used by COVID cases

→

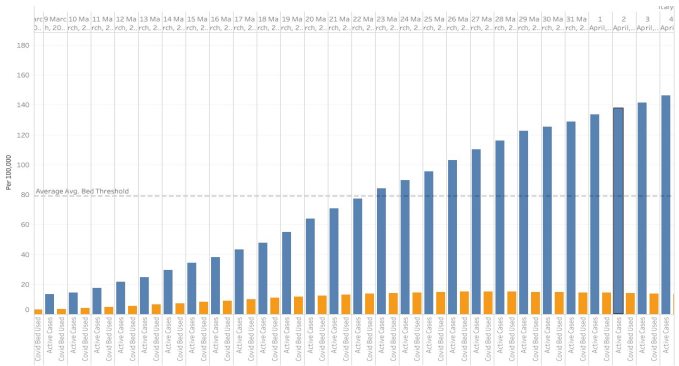
- Available beds per day
  - Per population
  - normalized

EDA Summary-

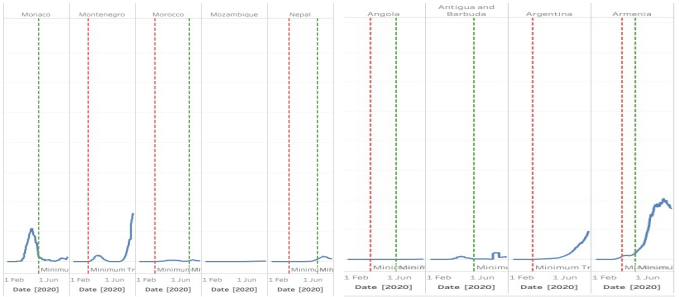
Challenges



- Checked for general trends in number of active cases
  - Cases go up starting mid-May, and for most countries decreases after the end of July
  - Italy seems to be an epicenter in Europe
  - United states continues to grow while most countries drop
  - Missing countries



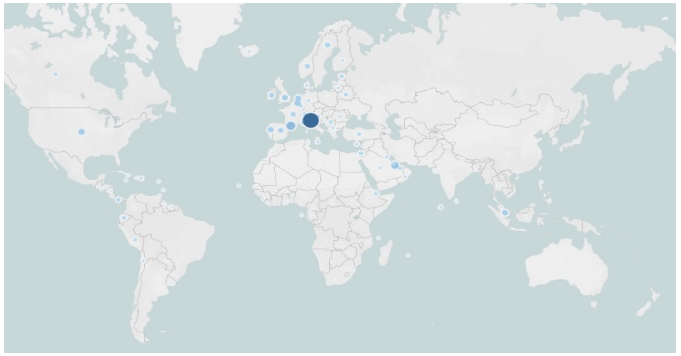
- Checking the severity of COVID hospitalization numbers
  - General trend as expected : as CVOID cases increase so does the rate of hospitalization.
  - Hospital capacity reference line: Why does it show that most countries are below their hospital capacity in terms of beds.



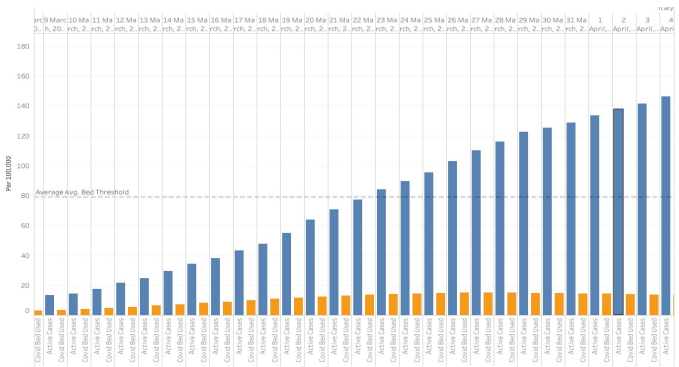
- Checking the significance of mandates
  - increase in COVID trends AFTER travel restrictions are put in place
  - decrease in COVID cases after travel restriction were lifted
  - Various trends

EDA Summary-

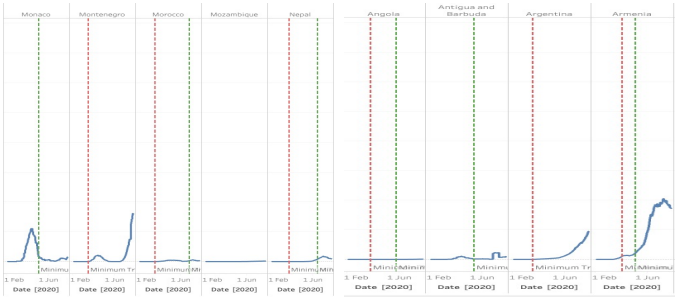
Solutions



- Missing countries
  - Unique categorical column
  - Removed



- Not enough examples of countries hitting o hospital bed capacity
  - Check model with regression scores first
  - If regression models do not perform well, Could be a classification problem.
  - Low/med/high



- Clarifying Trends
  - Binary to multinomial
  - [0,1] → [0,1,2]

Overview

1. Introducing four Models
2. Evaluate model performance for predicting number of available beds
    1. Target attribute = number of available beds
3. Evaluate model performance for predicting fatality rate
    1. Target attribute = rate of fatality
4. Pick the best model (1 out of 4)
5. Generate future data
6. Pass future data into our chosen model to predict future number of available beds per country for ***future dates***.
    1. Target attribute = number of available beds
7. Uses future data and future number of available beds (obtained in previous step) to predict future fatality rates.
    1. Target attribute = fatality rate
8. Compare how the number of available beds (capacity) affects the rate of fatality.

Linear Regression (base-line)

Random Forest

Neural Net – MLP

Neural Net - LSTM

What models are we trying?

Linear Regression	• Simplicity • Baseline model
Random Forest	• Less influenced by outliers • Can handle collinearity
Neural Net – MLP	• Can learn from faults • Used for complex problem solving
Neural Net – LSTM	• More complex than MLP • Better for time series data • Deals with vanishing gradient problem

Why did we choose these models?

Data Preparation – Part 1

- Splitting Data

```
features = df_raw.iloc[:, 0:-1]
labels = df_raw.iloc[:, [-1]]
print(features)
print(labels)

x_train, x_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, shuffle=True, random_state=42)

x_train, y_train = np.array(x_train), np.array(y_train)
x_test, y_test = np.array(x_test), np.array(y_test)
```

- Attributes used

Independent Attributes

- Day since first case
- Total test
- Active cases
- Travel mandate [0,1]
- Stay home mandate [0,1]
- Education mandate [0,1]
- Business closure mandate [0,1]
- Strict gathering mandate [0,1]

Target Attribute

- Available beds

- Linear Regression

```
# Logistic Regression
# https://scikit-learn.org/stable/modules/generated/sklearn.linea
from sklearn.linear_model import LinearRegression

linear_reg = LinearRegression(fit_intercept = False, n_jobs=-1)
linear_reg.fit(x_train, y_train)
y_pred_LR = linear_reg.predict(x_test)
```

- Random Forest

```
# Random Forest
# https://scikit-learn.org/stable/modules/generated

from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(random_state=42)
rf.fit(x_train, y_train)
y_pred_RF = rf.predict(x_test)
```

- Neural Net - MLP

```
# Neural Network - MLP
# https://scikit-learn.org/stable/modules/generated/sklea
from sklearn.neural_network import MLPRegressor

neural_MLP = MLPRegressor(random_state=42, max_iter=500)
neural_MLP.fit(x_train, y_train)
y_pred_MLP = neural_MLP.predict(x_test)
```

Training Models

- Neural Net - LSTM

```
# Neural Network - LSTM
# Tensorflow imports
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
from sklearn.metrics import r2_score

def init_LSTM(hidden_size, input_length, output_length, activation='relu', optimizer='adam', loss='mean_squared_error'):
  model = Sequential()
  model.add(LSTM(units=hidden_size, return_sequences=True, input_shape=(1, input_length)))
  model.add(LSTM(units=hidden_size, return_sequences=True))
  model.add(LSTM(units=hidden_size))
  model.add(Dense(units=output_length, activation=activation))
  model.compile(optimizer=optimizer, loss=loss)

  model.score=r2_score

  return model

x_train_LSTM = np.reshape(x_train, (-1,1, x_train.shape[1]))
x_test_LSTM = np.reshape(x_test, (-1,1, x_test.shape[1]))

lstm_model = init_LSTM(50,x_train_LSTM.shape[2], y_train.shape[1])
lstm_model.fit(x_train_LSTM, y_train, epochs=30, batch_size=50, validation_split=.05)
y_pred_LSTM = lstm_model.predict(x_test_LSTM)
```

Training models continued

Evaluations - Code

- Statistical scores
  - 1. R squared
  - 2. Mean squared error
  - 3. Root mean squared error
  - 4. mean absolute error

```
y_test_shifted = y_test * 1000
y_pred_LR_shifted = y_pred_LR * 1000

print(f'R squared: {linear_reg.score(x_test, y_test)}')
print()
print('Mean squared error: %.2f'% mean_squared_error(y_test_shifted, y_pred_LR_shifted))
print('Root Mean squared error: %.2f'% mean_squared_error(y_test_shifted, y_pred_LR_shifted) ** 0.5)
print('Mean absolute error: %.2f'% mean_absolute_error(y_test_shifted, y_pred_LR_shifted))
print()
```

- Actual vs predicted graph

```
x=range(len(x_test))
plt.scatter(x,y_test_shifted, label = 'Actual')
plt.scatter(x,y_pred_LR_shifted,label = 'Predicted')
plt.xlabel('Observations')
plt.legend()
plt.show()
```

- Residual error graph

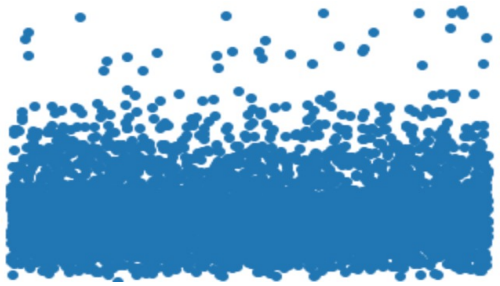
```
g=plt.plot(y_test_shifted - y_pred_LR_shifted,marker='o',linestyle='')
```

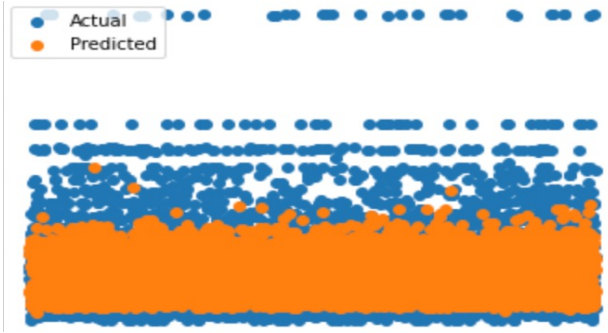
- Actual vs predicted graph bar graph

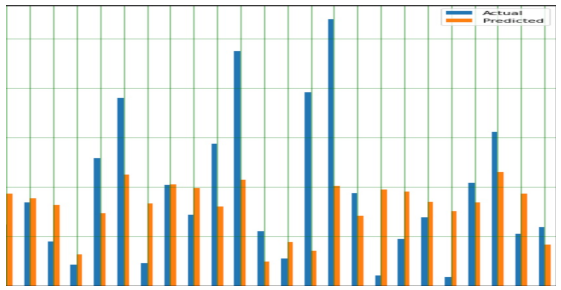
```
df4 = pd.DataFrame({'Actual': y_test_shifted.flatten(), 'Predicted': y_pred_LR_shifted.flatten()})
df4
df5 = df4.head(50)
df5.plot(kind='bar',figsize=(16,10))
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
plt.show()
```

Linear Regression

```
R squared: -0.20615730878588945

Mean squared error: 4935.69
Root Mean squared error: 70.25
Mean absolute error: 50.72
```

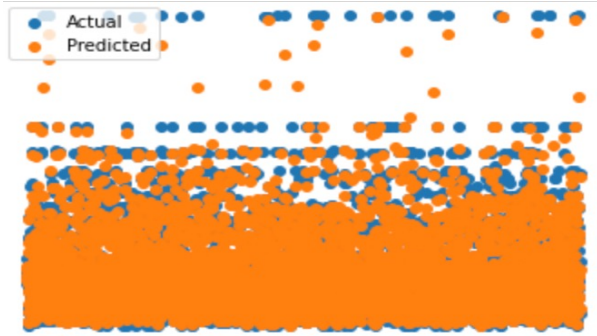


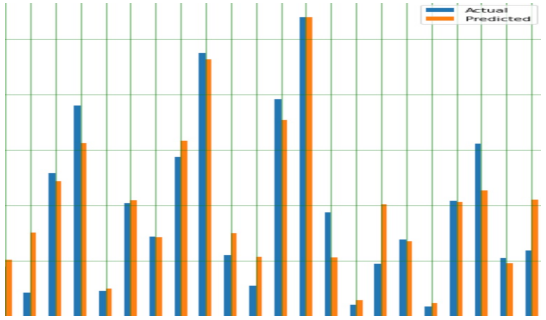




Random Forest

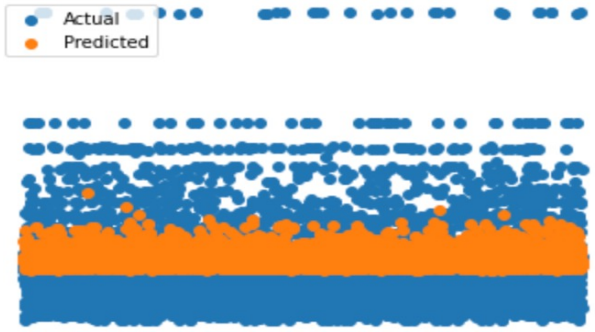
```
R squared: 0.7084231974720095

Mean squared error: 1193.16
Root Mean squared error: 34.54
Mean absolute error: 17.59
```

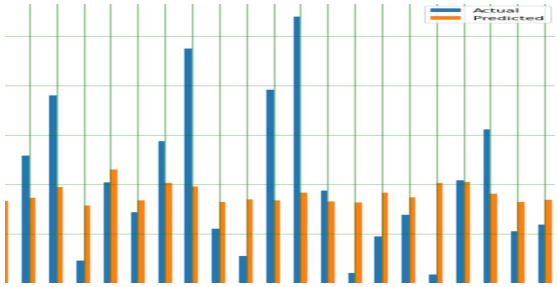




Neural Net - MLP

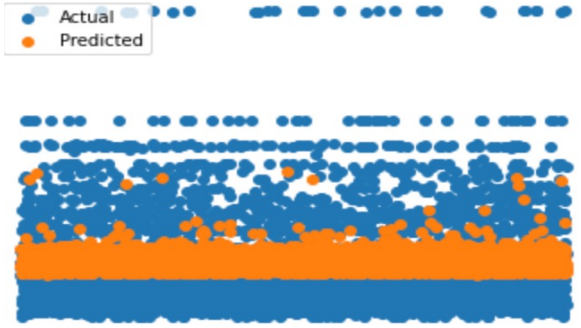
```
R squared: 0.012457692629307404

Mean squared error: 4041.10
Root Mean squared error: 63.57
Mean absolute error: 49.64
```

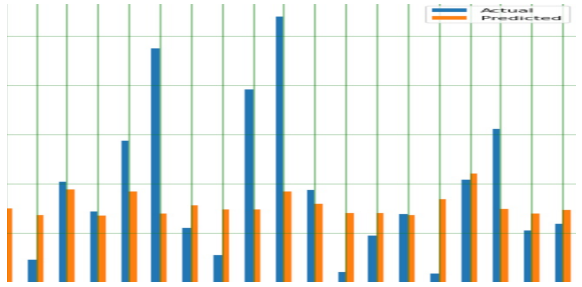




Neural Net - LSTM

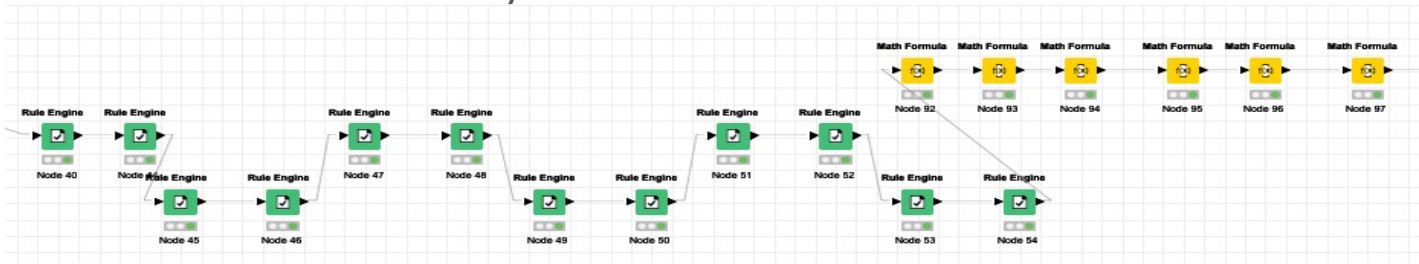
```
R squared: 0.043078714111582594

Mean squared error: 3915.80
Root Mean squared error: 62.58
Mean absolute error: 45.66
```





Data Preparation – Part 2

- Multinomial from Binary



- Sliding Window

```
def create_window(df_org,target_attr_name):
  df = df_org.copy()
  df_raw_sliding_window = pd.DataFrame()

  for name,group in df.groupby('Country Name'):
    #print(f"Generating sliding window for {name}")
    df_countrylevel = pd.DataFrame()

    target_attr_col = pd.DataFrame(group[target_attr_name].values)
    df_sliding_target_attr = pd.concat([target_attr_col.shift(5),target_attr_col.shift(4),
                                        target_attr_col.shift(3),target_attr_col.shift(2),
                                        target_attr_col.shift(1),target_attr_col], axis=1)
    df_sliding_target_attr.columns = ['t-5','t-4','t-3','t-2','t-1',target_attr_name]

    df_left = group.drop([target_attr_name], axis=1).reset_index(drop=True)
    df_countrylevel = pd.concat([df_left, df_sliding_target_attr], axis=1)

    df_raw_sliding_window = pd.concat([df_raw_sliding_window,df_countrylevel], ignore_index=True)


  return df_raw_sliding_window.dropna(subset=['t-5','t-4','t-3','t-2','t-1'],inplace=False)
```

Data Preparation – Part 2

- Splitting Data

```
features = df_raw.iloc[:, 0:-1]
labels = df_raw.iloc[:, [-1]]
print(features)
print(labels)

x_train, x_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, shuffle=True, random_state=42)

x_train, y_train = np.array(x_train), np.array(y_train)
x_test, y_test = np.array(x_test), np.array(y_test)
```

- Attributes used

Independent Attributes

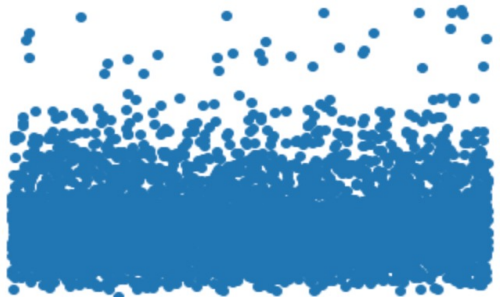
- Day since first case
- Total test
- Active cases
- Travel mandate [0,1,2]
- Stay home mandate [0,1,2]
- Education mandate [0,1,2]
- Business closure mandate [0,1,2]
- Strict gathering mandate [0,1,2]
- Sliding window (adds five columns)
  - Based on available beds

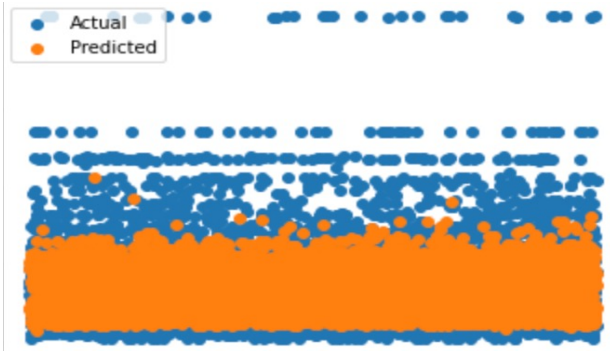
Target Attribute

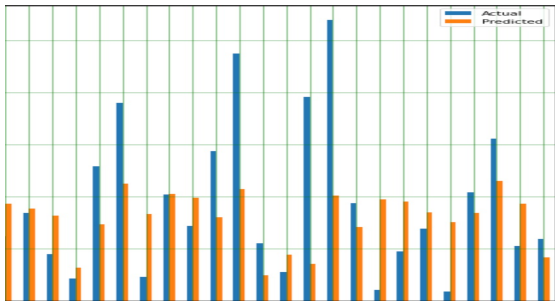
- Available beds

Linear Regression [old data]

```
R squared: -0.20615730878588945

Mean squared error: 4935.69
Root Mean squared error: 70.25
Mean absolute error: 50.72
```

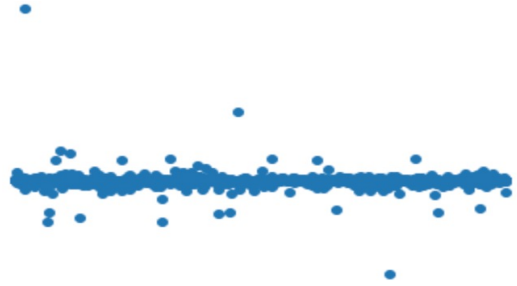


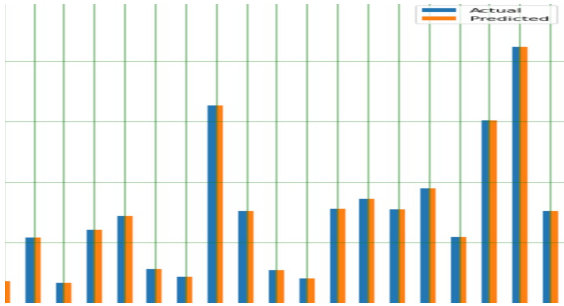




Linear Regression [new data]

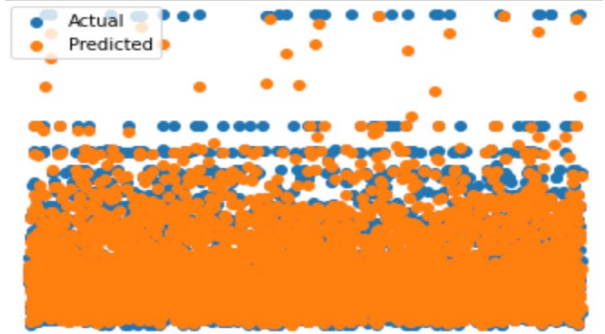
```
R squared: 0.9999992501520714

Mean squared error: 0.00
Root Mean squared error: 0.05
Mean absolute error: 0.01
```

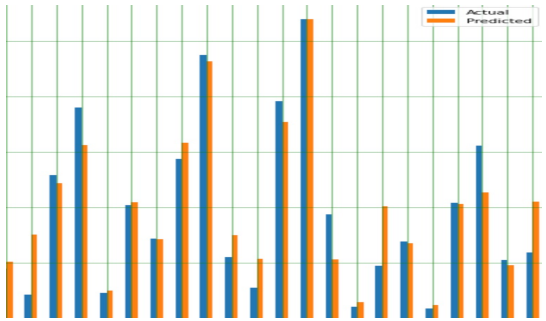




Random Forest [old data]

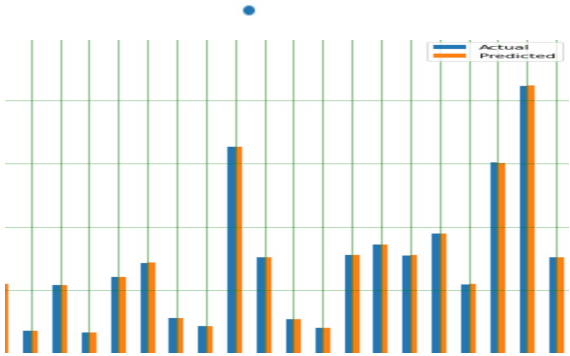
```
R squared: 0.7084231974720095

Mean squared error: 1193.16
Root Mean squared error: 34.54
Mean absolute error: 17.59
```





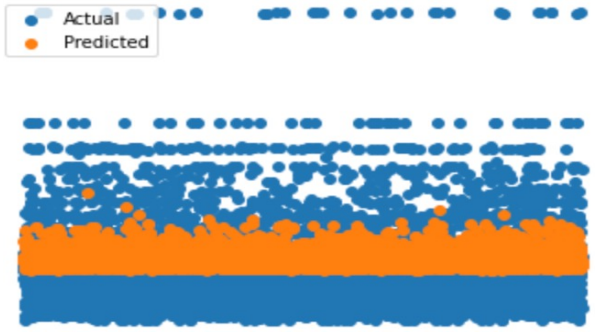
Random Forest [new data]

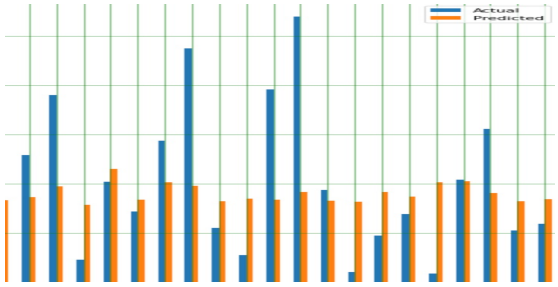
```
R squared: 0.999975517565039

Mean squared error: 0.10
Root Mean squared error: 0.31
Mean absolute error: 0.13
```



Neural Net – MLP [old data]

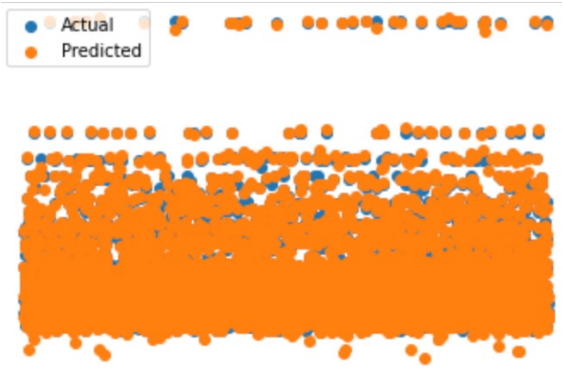
```
R squared: 0.012457692629307404

Mean squared error: 4041.10
Root Mean squared error: 63.57
Mean absolute error: 49.64
```

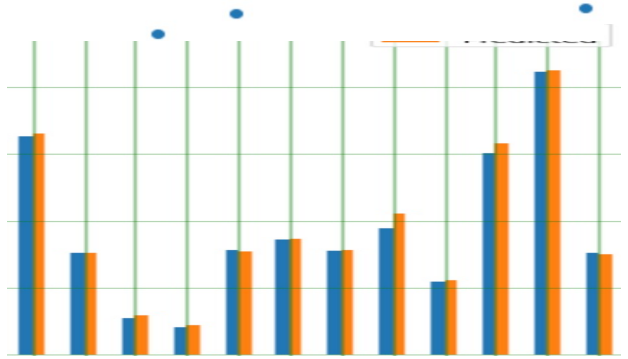




Neural Net – MLP[new data]

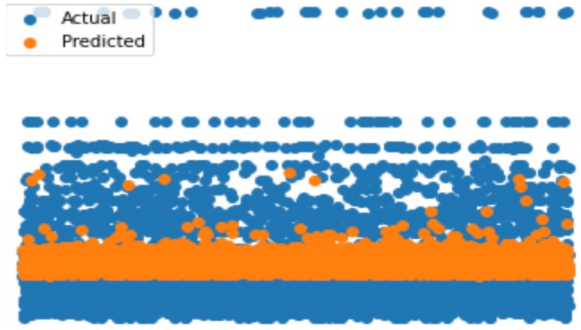
```
R squared: 0.9902413964967762

Mean squared error: 38.94
Root Mean squared error: 6.24
Mean absolute error: 3.30
```

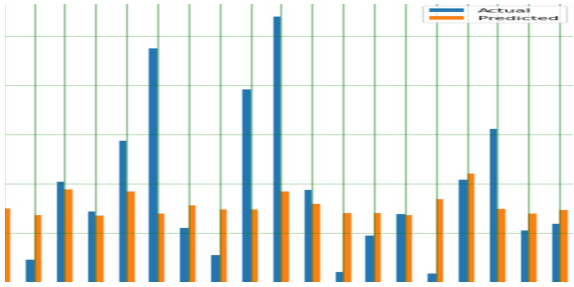




Neural Net – LSTM [old data]

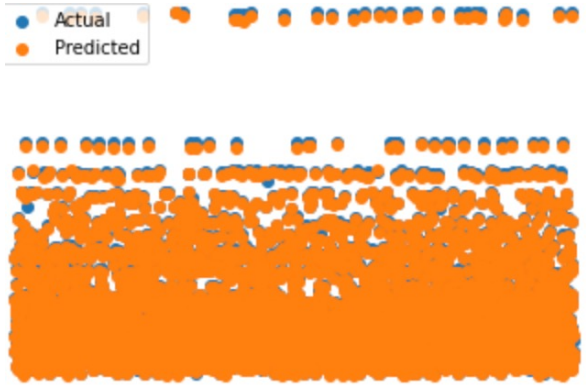
```
R squared: 0.043078714111582594

Mean squared error: 3915.80
Root Mean squared error: 62.58
Mean absolute error: 45.66
```

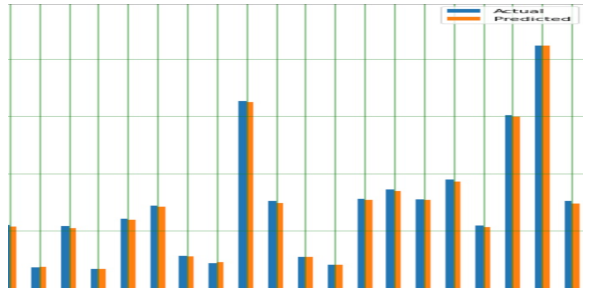




Neural Net – LSTM [new data]

```
R squared: 0.9990131888193795

Mean squared error: 3.94
Root Mean squared error: 1.98
Mean absolute error: 1.57
```





Data Preparation – Part 2

- Attributes used

Independent Attributes

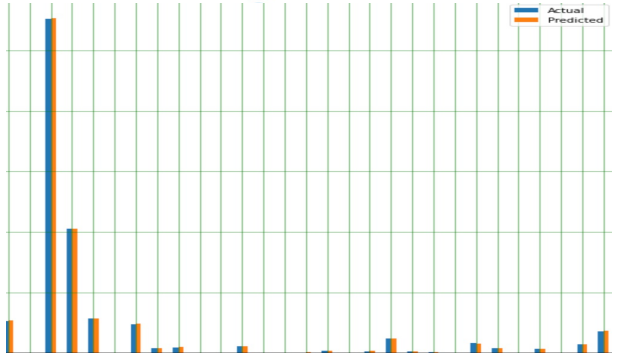
- Day since first case
- Total test
- Active cases
- Available beds
- Travel mandate [0,1,2]
- Stay home mandate [0,1,2]
- Education mandate [0,1,2]
- Business closure mandate [0,1,2]
- Strict gathering mandate [0,1,2]
- Sliding window (adds five columns)
  - Based on fatality rate

Target Attribute

- Fatality rate

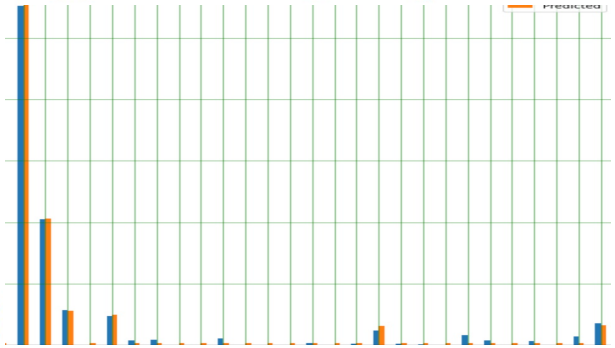
Linear Regression

```
R squared: 0.9997731135978736

Mean squared error: 0.05
Root Mean squared error: 0.22
Mean absolute error: 0.04
```



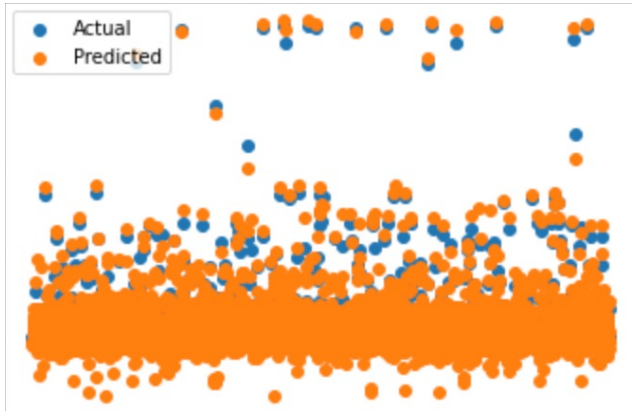
Random Forest

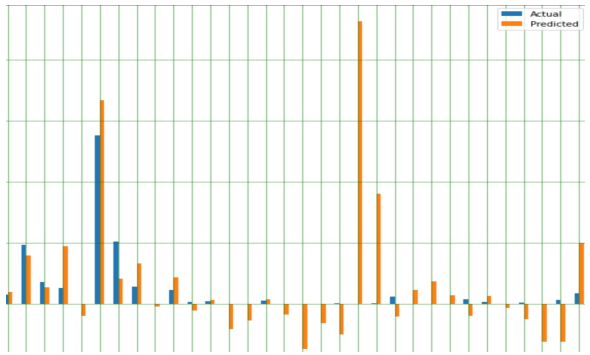
```
R squared: 0.9994296299970932

Mean squared error: 0.13
Root Mean squared error: 0.35
Mean absolute error: 0.22
```



Neural Net - MLP

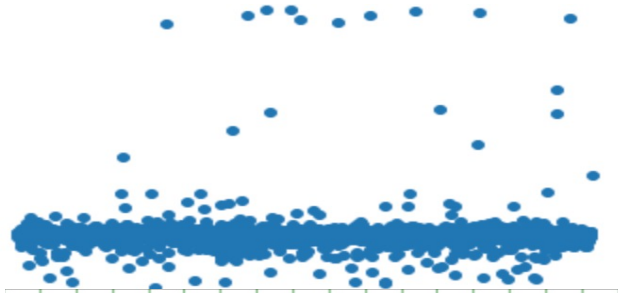
```
R squared: 0.7362190421332636

Mean squared error: 57.82
Root Mean squared error: 7.60
Mean absolute error: 4.91
```

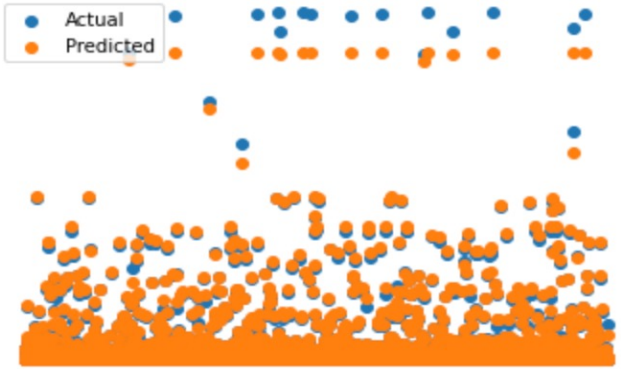


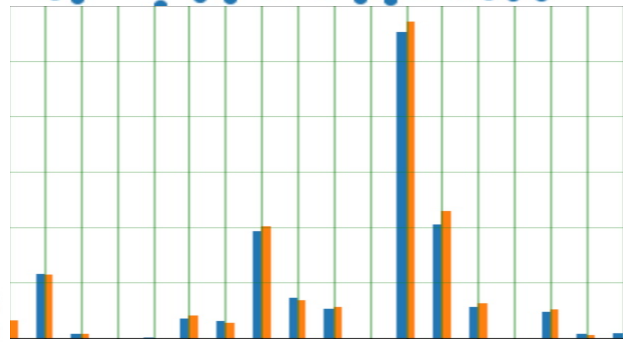


Neural Net – LSTM

```
R squared: 0.9935853271607126

Mean squared error: 1.41
Root Mean squared error: 1.19
Mean absolute error: 0.30
```







Best Model?

Linear Regression

Random Forest

Neural Net – MLP

Neural Net - LSTM

Generating future data

```
for name,group in df_raw.groupby('Country/Region'):

    df_countrylevel = pd.DataFrame()

    date_col = pd.DataFrame(group['Date'].values)
    target_attr_col1 = pd.DataFrame(group['Sum(Confirmed)'].values)
    target_attr_col2 = pd.DataFrame(group['Sum(Deaths)'].values)
    target_attr_col3 = pd.DataFrame(group['Sum(Recoveries)'].values)

    df = pd.concat([date_col,target_attr_col1],axis=1)
    df.columns = ['ds','y']
    predictor = Prophet(changepoint_prior_scale=0.15)
    predictor.fit(df)
    forecast_confirmed = predictor.make_future_dataframe(periods=365, freq='D')
    forecast_confirmed = predictor.predict(forecast_confirmed)
    forecast_confirmed = forecast_confirmed[['ds','trend']]

    df = pd.concat([date_col,target_attr_col2],axis=1)
    df.columns = ['ds','y']
    predictor = Prophet(changepoint_prior_scale=0.15)
    predictor.fit(df)
    forecast_death = predictor.make_future_dataframe(periods=365, freq='D')
    forecast_death = predictor.predict(forecast_death)
    forecast_death = forecast_death[['ds','trend']]

    df = pd.concat([date_col,target_attr_col3],axis=1)
    df.columns = ['ds','y']
    predictor = Prophet(changepoint_prior_scale=0.15)
    predictor.fit(df)
    forecast_recoveries = predictor.make_future_dataframe(periods=365, freq='D')
    forecast_recoveries = predictor.predict(forecast_recoveries)
    forecast_recoveries = forecast_recoveries[['ds','trend']]


    df_countrylevel = pd.merge(forecast_confirmed,forecast_death,on="ds")
    df_countrylevel = pd.merge(df_countrylevel,forecast_recoveries,on="ds")
    df_countrylevel['Country/Region'] = name

    df_raw_forecast = pd.concat([df_raw_forecast,df_countrylevel], ignore_index=True)
    print(df_raw_forecast.tail())
    print(df_raw_forecast.info)

df_raw_forecast.columns = ['ds','Forecast(Confirmed)','Forecast(Deaths)','Forecast(Recoveries)','Country/Region']
df_raw_forecast.to_csv(f"{home}/Desktop/BMIF4-forecasted.csv",index=False)
```

- Why do we need to generate future data?
  - We can pass future attribute values into our best model and predict future numbers of our target attribute, which right now, is number of available beds .
  - Future data generated from past data
  - Future data generated till November 2020
- Module used : *fbprophet*

When will each country run out of hospital beds?

- Train model using ***past available beds*** data

```
# Logistic Regression to predict future number of available beds trained
# https://scikit-learn.org/stable/modules/generated/sklearn.linear_model
from sklearn.linear_model import LinearRegression

linear_reg_future = LinearRegression(fit_intercept = False, n_jobs=-1)
linear_reg_future.fit(x_beds_train, y_beds_train)
```

- Now instead of testing models using test data, we now test model using future data to obtain future number of available beds

```
features = df_beds.iloc[:, 0:-1]
labels = df_beds.iloc[:, [-1]]
x_beds_train, _, y_beds_train, _ = train_test_split(features, labels, test_size=0.2, shuffle=True, random_state=42)
x_beds_train, y_beds_train = np.array(x_beds_train), np.array(y_beds_train)
```

```
# future dataset is now our test data
x_beds_future = np.array(df_beds_future)
```

- Now we have future number of available hospital beds

When will low hospital bed capacity affect rate of fatality?

- Now we want to predict future fatality numbers based on our new hospital bed numbers.
- Train model using ***past fatality rate*** data

```
# Logistic Regression to predict future number of fatality rate trained
# https://scikit-learn.org/stable/modules/generated/sklearn.linear_mode
from sklearn.linear_model import LinearRegression

linear_reg_future = LinearRegression(fit_intercept = False, n_jobs=-1)
linear_reg_future.fit(x_fat_train, y_fat_train)
```

- Append future available hospital beds to future data being its used as test data because now our target attribute is fatality rate and available beds becomes an independent attribute.

```
#update fat df and test set with available beds inserted at same pos as org dataset
beds_idx_pos = df_fat.columns.get_loc('available_beds')
df_fat_future.insert(loc=beds_idx_pos, column='available_beds', value=y_pred_beds_future)
x_fat_future = np.array(df_fat_future)
```

- Now instead of testing models using test data, we now test model using future data to obtain future number of fatality rates

```
y_pred_fat_future = linear_reg_future.predict(x_fat_future)
```

- Now we have future number of fatality rates

Workflow clarified

Dataset 1 (bad data)

Models

LR RF MLP LSTM

Bad performance

Cannot predict available beds and fatality well

Added new attributes

Dataset 2 (good data)

Models

LR RF MLP LSTM

good performance

Can predict available beds and fatality well

Best model

Generate future data

Use future data to predict number of beds available in the future

Append future values of available beds as an independent attribute

Predict future fatality using future number of available beds

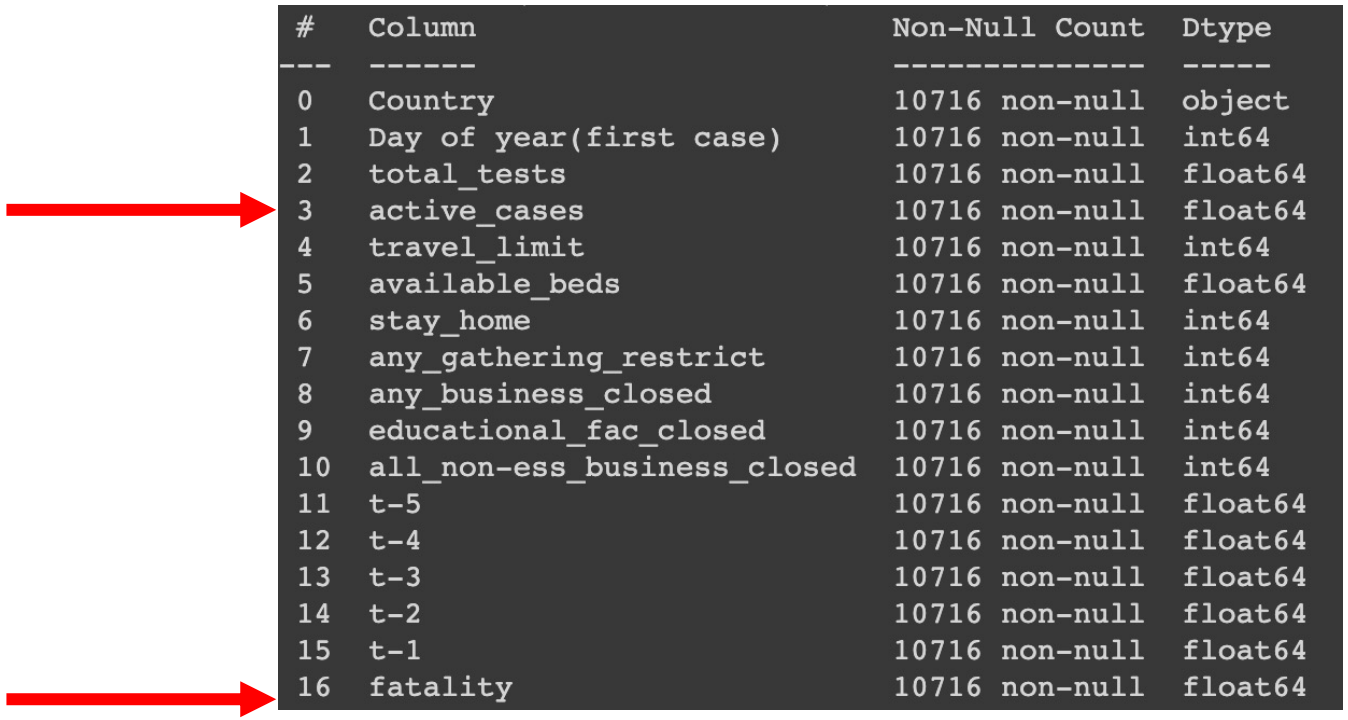
Compare number of available beds with fatality rate

What does our final data set look like ?

- Since we took country out before training models, first we need to add it back in

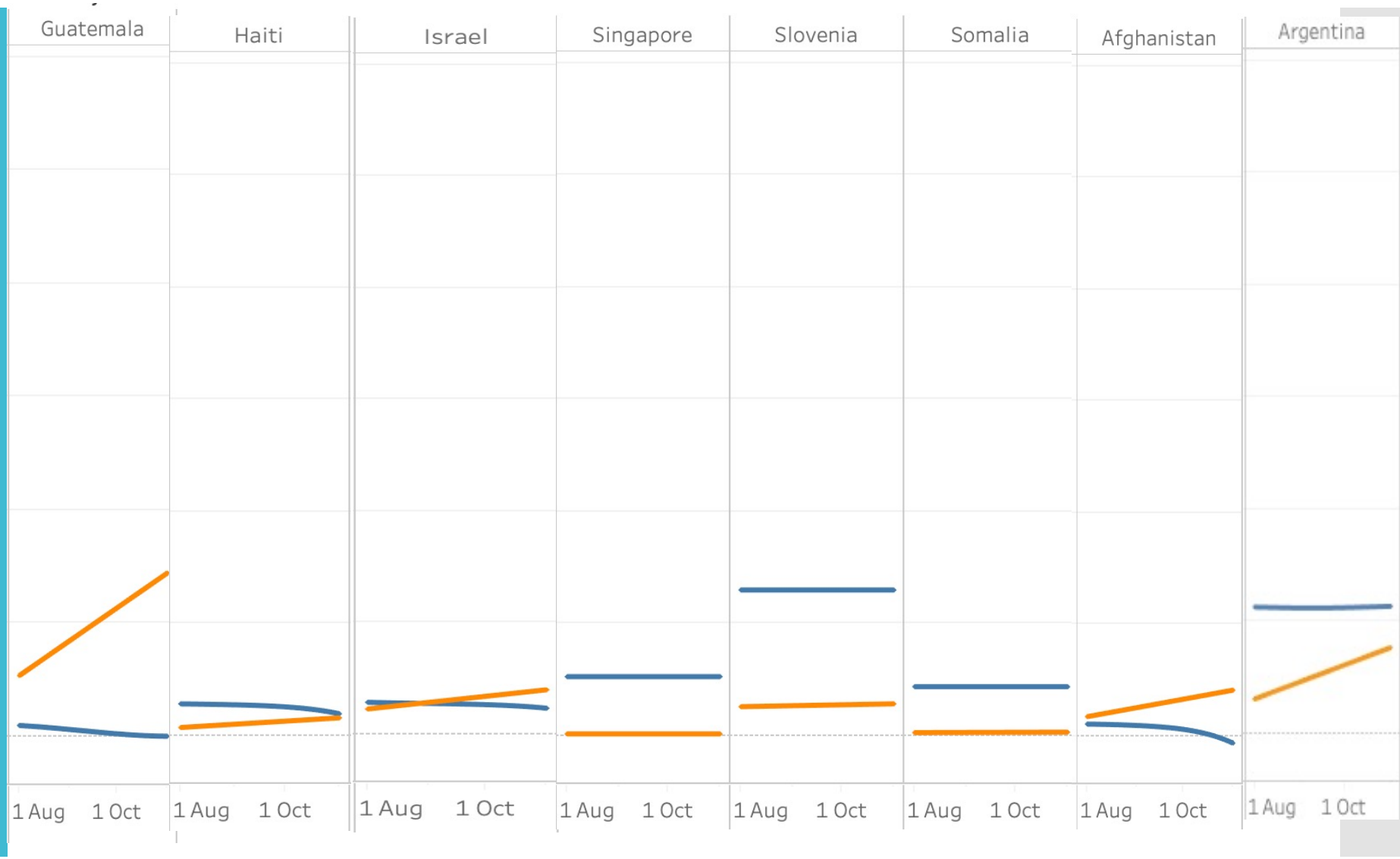
```
df_fat_future.insert(loc=0, column='Country',value=df_country_col)
```

- Quick look into the file



```
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Country                       10716 non-null  object
 1   Day of year(first case)       10716 non-null  int64
 2   total_tests                   10716 non-null  float64
 3   active_cases                  10716 non-null  float64
 4   travel_limit                  10716 non-null  int64
 5   available_beds                10716 non-null  float64
 6   stay_home                     10716 non-null  int64
 7   any_gathering_restrict        10716 non-null  int64
 8   any_business_closed           10716 non-null  int64
 9   educational_fac_closed        10716 non-null  int64
 10  all_non-ess_business_closed   10716 non-null  int64
 11  t-5                           10716 non-null  float64
 12  t-4                           10716 non-null  float64
 13  t-3                           10716 non-null  float64
 14  t-2                           10716 non-null  float64
 15  t-1                           10716 non-null  float64
 16  fatality                      10716 non-null  float64
```

How does hospital bed capacity affect rate of fatality ?

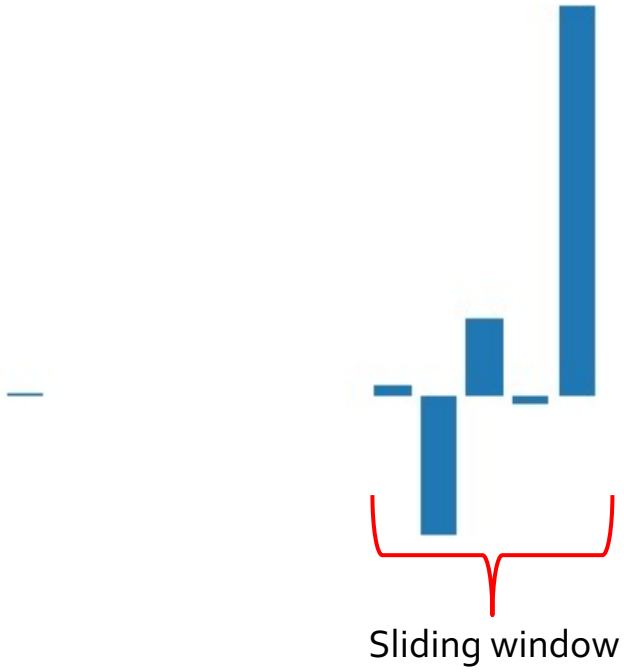


Why good models, but bad results ?

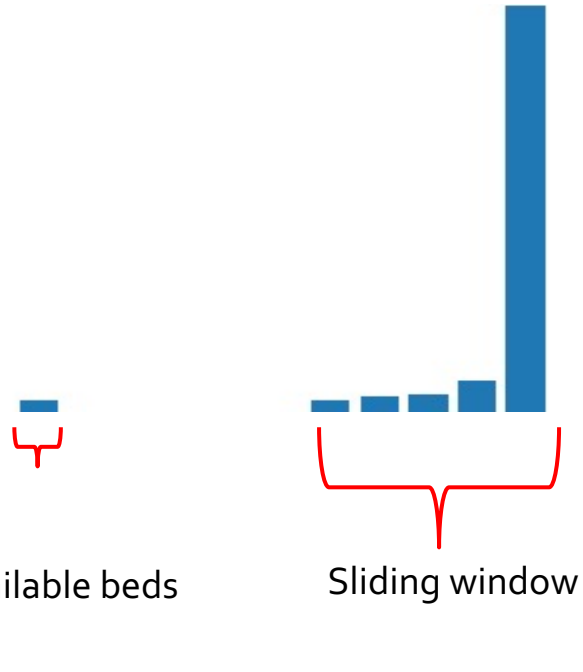
- If our models gave us good performance scores, why can't we see a relationship between rate of fatality and number of available beds?

- Feature importance

- Linear Regression



- Random Forest



Future Work

- K- fold validation
- Adding significant attributes
- Longer time frame for future data
- Try more regression models
- Try switching to a classification problem
- Try with province/state level data

Resources Used

- Related articles used to evaluate our models / results
  - https://towardsdatascience.com/how-to-evaluate-your-machine-learning-models-with-python-code-5f8d2d8d945b
  - https://towardsdatascience.com/a-beginners-guide-to-linear-regression-in-python-with-scikit-learn-83a8f7ae2b4f
  - https://towardsdatascience.com/time-series-analysis-in-python-an-introduction-70d5a5b1d52a
  - https://statisticsbyjim.com/regression/overfitting-regression-models/
- Data sources
  - http://www.healthdata.org/covid/data-downloads
  - https://data.humdata.org/dataset/novel-coronavirus-2019-ncov-cases
  - https://data.worldbank.org/indicator/SP.POP.TOTL
- Data preparation and statistics
  - KNIME
  - Python (sliding window)
- Data visualization
  - Tableau
  - Python
- Modeling
  - Python (google Colab)

Thank you for listening