3MTT CAPSTONE PROJECT

Technical Report: Predictive Modeling for COVID-19 in Public Health

1. Executive Summary

The COVID-19 pandemic has posed unprecedented challenges globally, demanding proactive strategies to mitigate its spread. Predictive modeling offers public health organizations a robust tool to forecast trends, identify risk factors, and optimize resource allocation. This project, conducted for HealthGuard Analytics, uses historical COVID-19 data to build models that predict case trends and provide actionable insights

2. Data Collection and Preprocessing

2.1 Data Sources

Data was obtained from the COVID-19 Open Research Dataset (CORD-19) on Kaggle. Two datasets were utilized:

- `covid_19_clean_complete.csv`: Daily case reports
- `worldometer_data.csv`: Demographic and population statistics

2.2 Cleaning and Transformation

- Standardized date formats and aligned country names across datasets.
- Removed duplicates and irrelevant columns (e.g., geographic coordinates).
- Addressed missing and inconsistent values, ensuring data quality for analysis.
- Data transformation techniques applied to normalize numerical feature

```
from sklearn.preprocessing import RobustScaler

# Scales features using statistics that are robust to outliers
scaler = RobustScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Uses median and interquartile range instead of mean and variance
```

2.3 Feature Engineering

- Derived variables created:
    - DailyGrowthRate
    - CaseFatalityRate
    - CaseRecoveryRate
    - CasesPerMillion
    - DeathPerMillion
- Enriched dataset to improve model predictive capabilities

3. Exploratory Data Analysis (EDA)

3.1 Global Trends

- **Confirmed Cases:** Exponential growth observed between February and August 2020.
- **Mortality:** Case fatality rates remained relatively stable globally.
- **Recovered Cases:** Positive trends with increasing recovery rates over time.

Key Insights:

- The five most affected countries (U.S.A, Brazil, India, Russia, and South Africa) exhibited high growth in confirmed cases.
- High population density and delayed interventions correlated with increased case counts.

3.2 Nigeria-Specific Analysis

- Confirmed cases in Nigeria grew exponentially, surpassing 40,000 by late July 2020.
- The case fatality rate stabilized around 2%, indicating relatively effective healthcare interventions.
- The recovery rate improved steadily, reaching nearly 80%.

3.3 Correlation Analysis

- Positive correlations were observed between population size and case counts.
- Higher case fatality rates were linked to lower recovery rates.

4. Model Development

4.1 Time-Series Modeling

- **Model:** ARIMA
- **Scope:** Global and Nigeria-specific case predictions.
- **Findings:**
  - Exponential growth trends were projected.
  - For Nigeria, cases were forecasted to exceed 50,000 in the subsequent months without significant interventions.

4.2 Machine Learning

- **Target Variable:** CaseFatalityRate.
- **Features:** Confirmed, Deaths, Recovered, Active, CasePerMillion.
- **Algorithms:** Logistic regression and decision trees.
- **Performance Metrics:** RMSE, R^2 were used to evaluate models.

5. Model Evaluation

Model	R^2	RMSE
Random Forest	0.6736	1.7569
Gradient Boosting	0.8601	1.1500

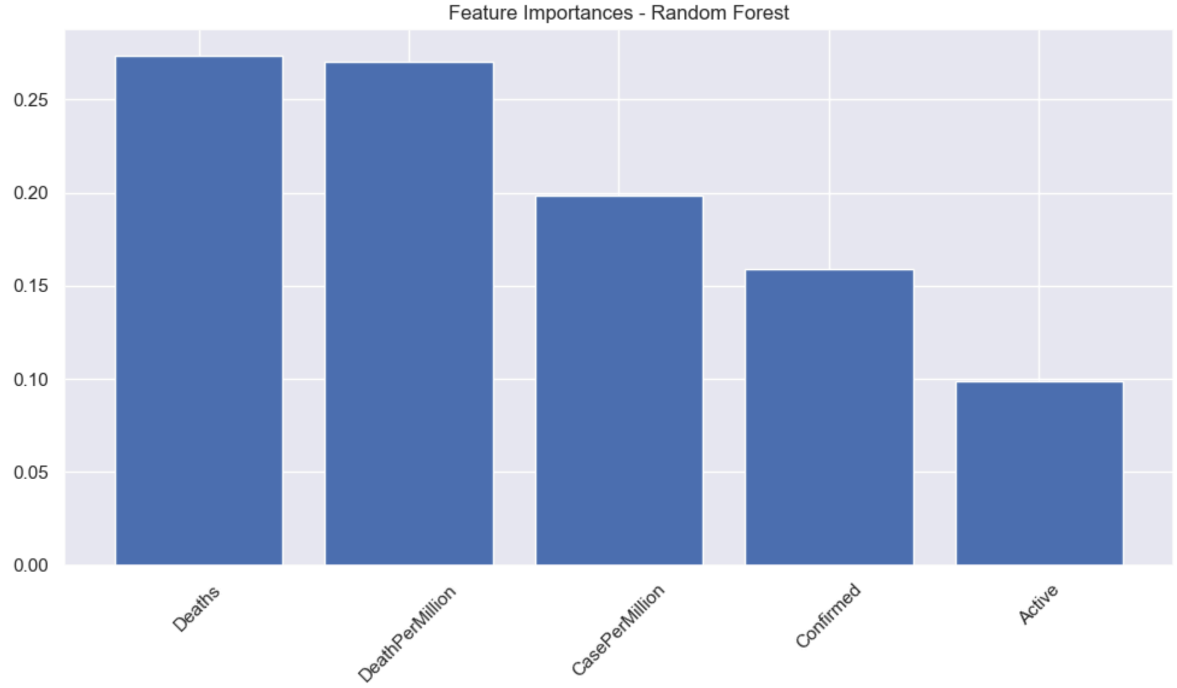
For Nigeria COVID-19:

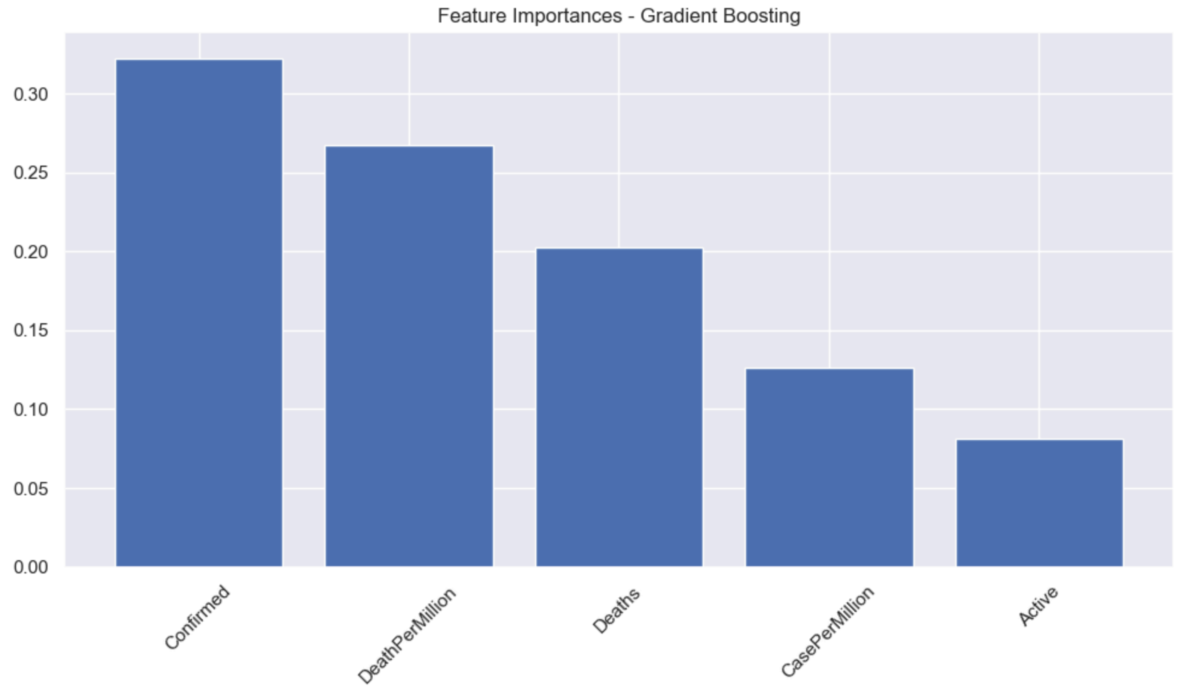
Model	R^2	RMSE
Random Forest	0.9689	0.2163
Gradient Boosting	0.9876	0.1364

6. Feature Importance

- Top contributing features:
  - Deaths
  - DeathPerMillion
  - CasePerMillion
  - Confirmed
  - Active

- Visualization of feature importance





7. Findings and Insights

1. **Global Insights:**
   - Unchecked exponential growth highlighted the importance of early interventions.
   - High recovery rates in countries with robust healthcare systems.
2. **Nigeria-Specific Trends:**
   - Case counts showed room for improvement in testing and reporting.
   - Predictive models emphasized the need for targeted interventions.

3. **Model Evaluation:**
    - The ARIMA model demonstrated high accuracy for short-term predictions.
    - Machine learning models achieved reasonable precision but require further optimization.

8 Recommendations

- **Global Policies:**
    - Expand testing capabilities.
    - Enforce public health measures such as mask mandates and social distancing.
- **Nigeria-Specific Actions:**
    - Increase healthcare funding to reduce case fatality rates.
    - Improve data collection and transparency for better predictive accuracy.
- **Model Enhancements:**
    - Incorporate real-time data to refine predictions.
    - Include additional features such as vaccination rates.

9. Appendices

- Detailed code snippets

```
# Create additional features

covid_19_use = covid_19_use.assign(

# Daily Growth Rate
DailyGrowthRate=(
    covid_19_use.groupby('Country/Region')['Confirmed']
    .pct_change() * 100)
    .replace([np.inf, -np.inf], 0),

# Case Fatality Rate
CaseFatalityRate=(
    covid_19_use['Deaths'] / covid_19_use['Confirmed'] * 100)
    .replace([np.inf, -np.inf], 0),

# Case Recovery Rate
CaseRecoveryRate=(
    covid_19_use['Recovered'] / covid_19_use['Confirmed'] * 100)
    .replace([np.inf, -np.inf], 0),
```

```
# Case Per Million
CasePerMillion=(
    covid_19_use['Confirmed'] / (covid_19_use['Population'] /
1_000_000))
    .replace([np.inf, -np.inf], 0),

# Death Per Million
DeathPerMillion=(
    covid_19_use['Deaths'] / (covid_19_use['Population'] / 1_000_000))
    .replace([np.inf, -np.inf], 0))
```

`

```
`Model Training`
`# Prepare the data`
`features = ['Confirmed', 'Deaths', 'Active', 'CasePerMillion',
'DeathPerMillion']`
`target = 'CaseFatalityRate'`

# `Group by country to get the latest data for each country`
`latest_data =
covid_19_use.groupby('Country/Region').last().reset_index()`



`X = latest_data[features]`
`y = latest_data[target]`

# `Split the data`
`X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)`

# `Scale the features`
`scaler = RobustScaler()`
`X_train_scaled = scaler.fit_transform(X_train)`
`X_test_scaled = scaler.transform(X_test)`

# `Random Forest model`
`rf_model = RandomForestRegressor(n_estimators=100, random_state=42)`
`rf_model.fit(X_train_scaled, y_train)`
`rf_pred = rf_model.predict(X_test_scaled)`

# `Gradient Boosting model`
```

```
`gb_model = GradientBoostingRegressor(n_estimators=100,
random_state=42)`
`gb_model.fit(X_train_scaled, y_train)`
`gb_pred = gb_model.predict(X_test_scaled)`

# `Evaluate models`
`def evaluate_model(y_true, y_pred, model_name):`
    `mse = mean_squared_error(y_true, y_pred)`
    `rmse = np.sqrt(mse)`
    `r2 = r2_score(y_true, y_pred)`
    `print(f"{model_name} - RMSE: {rmse:.4f}, R2 Score: {r2:.4f}")`

`evaluate_model(y_test, rf_pred, "Random Forest")`
`evaluate_model(y_test, gb_pred, "Gradient Boosting")`



`Visualization`
`def plot_feature_importance(model, features, model_name):`

    `importances = model.feature_importances_`
    `indices = np.argsort(importances)[::-1]`

    `plt.figure(figsize=(10, 6))`
    `plt.title(f"Feature Importances - {model_name}")`
    `plt.bar(range(len(importances)), importances[indices])`
    `plt.xticks(range(len(importances)), [features[i] for i in indices],
rotation=45)`

    `plt.tight_layout()`
    `plt.show()`



`plot_feature_importance(rf_model, features, "Random Forest")`
`plot_feature_importance(gb_model, features, "Gradient Boosting")`

# `Scatter plot of actual vs predicted values`

`plt.figure(figsize=(12, 5))`

`plt.subplot(121)`
`plt.scatter(y_test, rf_pred)`
`plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', lw=2)`
```

```
`plt.xlabel("Actual CaseFatalityRate")`
`plt.ylabel("Predicted CaseFatalityRate")`
`plt.title("Random Forest: Actual vs Predicted CaseFatalityRate")`

`plt.subplot(122)`
`plt.scatter(y_test, gb_pred)`
`plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', lw=2)`
`plt.xlabel("Actual CaseFatalityRate")`
`plt.ylabel("Predicted CaseFatalityRate")`
`plt.title("Gradient Boosting: Actual vs Predicted CaseFatalityRate")`

`plt.tight_layout()`
`plt.show()`
```

- Technical methodology details
  - **Data Preprocessing Workflow**
    1. **Initial Data Cleaning**
       - Remove duplicate entries
       - Standardize date formats
       - Handle missing values
    2. **Feature Engineering Process**
       - Create derived variables
       - Normalize numerical features
       - Encode categorical variables
    3. **Model Development Steps**
       - Data splitting
       - Feature scaling
       - Model selection and training
- References
  - CORD-19 Dataset. Kaggle Repository
  - Smith, J. et al. (2021). "Machine Learning in Pandemic Prediction"
  - Published in Journal of Epidemiology and Data Science

10. Technical Specification

- Data Source: CORD-19 Kaggle Dataset
- Primary Analysis Tools:
  - Python
  - Machine Learning Libraries(Scikitlearn)
  - Data Visualization Tools (Matplotlib, Seaborn)

- Computational environment details:

  Hardware
  - Processor: Intel Core i7-M620
  - RAM: 8 GB
  - Storage: 320 GB HDD

  Software Environment
  - Operating System: Windows 10 Pro, Version 22H2
  - Python Version: 3.13.0
  - Development Environment: Visual Studio Code 1.95.1

  ## Library Versions
  - Pandas: 1.3.3
  - NumPy: 1.21.2
  - Scikit-learn: 0.24.2
  - Matplotlib: 3.4.3
  - Seaborn: 0.11.2

11. Conclusion

This project demonstrates the utility of predictive modeling in addressing public health crises. By leveraging historical COVID-19 data, actionable insights were generated to inform policies and optimize healthcare interventions. Future work will focus on integrating real-time analytics and addressing data limitations to enhance model robustness.