Hybrid Forecasting of Sea Level Trends Using Classical and Quantum Machine Learning

PRAJAKTA GURAV

June 2025

1 Introduction

In the context of accelerating climate change, rising sea levels pose a significant challenge to coastal populations, infrastructure as well as the ecosystems. Forecasting these changes with precision has become not only a scientific priority but a necessity for sustainable policy planning. Traditional approaches, such as time series analysis, have been instrumental in modeling global mean sea level (GMSL), yet they often fall short in capturing trend dynamics or incorporating emerging computational challenges.

This chapter introduces SeaRiseIQ — a hybrid framework designed to explore both the numerical forecasting and directional classification of sea level trends using NASA's satellite-derived GMSL data till 2025. In addition to working with unadjusted values, we incorporate the Glacial Isostatic Adjustment (GIA), a correction that accounts for long-term vertical land motion and offers a more accurate representation of true ocean rise.

Our methodology combines autoregressive modeling (ARIMA) for forecasting with machine learning classifiers — Logistic Regression, Support Vector Machines, and a simulated Quantum Support Vector Machine (QSVM), to predict whether sea levels are rising or stabilizing in future years. By comparing adjusted and raw datasets across classical and quantum-inspired models, SeaRiseIQ provides a novel, layered perspective on the challenge of sea level prediction in a data-rich yet uncertain climate landscape.

This integration of quantum kernel methods into climate analytics, while still in its early stages, reflects a broader motivation: to explore the potential of hybrid computational techniques in environmental forecasting — not just for accuracy, but for adaptability, interpretability, and long-term resilience.

2 Background

2.1 Sea Level Rise and Climate Models

Sea level rise (SLR) is one of the most critical consequences of climate change, driven by the thermal expansion of oceans and accelerated melting of glaciers

and ice sheets. [1] This persistent increase in Global Mean Sea Level (GMSL) has far-reaching socioeconomic and environmental effects, particularly in low-lying and coastal regions. Forecasting GMSL has traditionally relied on statistical models like ARIMA, which use past observations to make future predictions. While these methods have proven effective for structured, linear trends, integrating more recent computational techniques including machine learning and quantum approaches — may offer deeper insights and improved trend detection in dynamic datasets.

Satellite-based data, particularly from NASA and NOAA, has made it possible to observe global mean sea level (GMSL) changes over time with high precision. However, these raw observations are affected by ongoing vertical shifts in the Earth's crust due to glacial rebound which is a phenomenon known as Glacial Isostatic Adjustment (GIA). Correcting for GIA provides a clearer picture of oceanic change, allowing scientists to isolate sea level variations caused by climate forces in detail rather than relying on tectonic motion alone.

2.2 Modeling Sea Level Changes: From Time Series to Quantum Learning

Traditional forecasting of sea level relies heavily on time series models such as the Autoregressive Integrated Moving Average (ARIMA). These models are particularly popular for such forecasting as they determine future values which are dependent on past observations and trends. ARIMA has been successfully applied to many environmental datasets due to its simplicity. However, it often has difficulty detecting sudden changes related to forecasting or identifying whether the trend is going up or down.

In recent years, machine learning (ML) approaches have gained popularity for environmental prediction tasks. Algorithms like Logistic Regression and Support Vector Machines (SVM) offer advantages in trend classification, especially when dealing with large, noisy, or non-linear data. These models can detect hidden patterns and offer flexible decision boundaries, which are beneficial in understanding short-term sea level dynamics.

As technology advances, there is growing interest in Quantum Machine Learning (QML), which combines ideas from quantum computing. Quantum kernel methods, such as the Quantum Support Vector Machine (QSVM), aim to project data into high-dimensional quantum spaces where complex relationships may be more easily separable. While practical implementation is still constrained by current hardware, simulated QML models offer a glimpse into the potential of quantum-enhanced forecasting systems.

This chapter builds on these foundations by integrating time series modeling, classical ML, and simulated quantum classifiers into a unified framework with the goal of capturing the trend of sea level changes using real-world satellite data.

3 Design and Implementation

This section outlines the workflow followed in the SeaRiseIQ framework, from raw data preparation to forecasting and trend classification using both classical and quantum-inspired approaches.

3.1 Dataset and Preprocessing

The data used in this project was sourced from NASA's Global Mean Sea Level (GMSL) archives, which provide sea level observations from satellite missions dating back to 1992. The original file was in .txt format and was converted to a structured CSV using a Python script for easier analysis and compatibility with machine learning tools.

The dataset contains two main measurements:

- GMSL_NoGIA: Global sea level without the Glacial Isostatic Adjustment (raw sea level)
- GMSL_GIA: GMSL adjusted for the Earth's crustal rebound due to melting glaciers.

To prepare the data for modeling:

- Missing values were handled appropriately.
- Lag features (values from previous time steps) were created to support supervised learning.
- A binary trend label was generated: 1 if the sea level increased from the previous time step, 0 if it stayed the same or decreased.

These labels allowed the dataset to be used in a classification setting in addition to time series forecasting.

To model and forecast sea level numerically, the ARIMA (AutoRegressive Integrated Moving Average) model was selected. This statistical model is widely used for time series analysis due to its ability to account for trends, seasonality, and noise in historical data. The "GMSL_GIA" column was chosen for forecasting as it reflects corrected and more reliable sea level trends. Parameters for the ARIMA model were selected based on autocorrelation and partial autocorrelation plots. Once trained, the model was used to predict future sea level values, along with 95% confidence intervals to account for uncertainty. In next step, Residual analysis was conducted to validate model accuracy. Residuals were visualized through Histograms (with overlaid normal curves), Time series residual plots, Outlier detection analysis.

Although ARIMA provided a reasonable forecast, it tended to produce flat-looking future curves when differencing was involved which is a known limitation of the model when recent data has low variation. In addition to numeric forecasting, we aimed to classify whether sea level would rise or not rise at the next step. For this, we used the following input features:

- GMSL_GIA
- GMSL_no_GIA

These were used to train the following models:

- Logistic Regression: A simple, interpretable classifier that served as a baseline.
- Support Vector Machine (SVM): Used with RBF kernel for handling non-linearity.
- Simulated Quantum Support Vector Machine (QSVM): A classical SVM mimicking quantum kernel behavior, useful in exploring QML potentials without real quantum hardware.

To handle class imbalance (more rising values than flat/falling ones), we used balanced training strategies. The performance of each model was evaluated using: Accuracy, Precision, Recall, F1 Score, Confusion matrices Among the models tested, Balanced SVM and Logistic Regression performed best in terms of fairness across both classes. QSVM produced competitive results and served as a proof-of-concept for using quantum inspired methods in Earth science.

4 Comparative Analysis

4.1 Forecasting Evaluation of ARIMA Performance

To understand the dynamics of sea level rise, we first analyzed NASA's Global Mean Sea Level (GMSL) dataset with and without Glacial Isostatic Adjustment (GIA). The dataset spans from 1992 to 2025 in monthly resolution. Our goal was twofold: (1) to forecast sea level using classical models, and (2) to classify trend directions using machine learning and quantum-inspired models.

4.2 Data Conversion

The original sea level data, obtained from the official NASA repository, was in a raw ASCII text format with headers and multiple columns of numerical values. To make it suitable for analysis, I programmatically converted it into a structured CSV format using Python. This involved downloading the file, filtering out non-data lines, assigning appropriate column names based on the dataset documentation, and saving it as a clean .csv file for further processing and visualization.

Listing 1: Data Conversion

```
import pandas as pd
import requests

# Download the file
```

```
url = "https://deotb6e7tfubr.cloudfront.net/s3-edaf5da92e0ce48fb61175c28b67e95d/podaac-ops-cumulus-protected.s3.us-west-2.amazonaws.com/MERGED_TP_J1_OSTM_OST_GMSL_ASCII_V52/GMSL_TPJAOS_5.2.txt?A-userid=strawberrykim&Expires=1749051153&Signature=gKu-dDRZ6V39exUvWOQ2VcErdPfHxG7rDhcKZ~~xOFbgYddfdEf2kWdrUn1KtcvJ7NHBISjKHTZy9dWgiiZssgGg41zk7BhczE-k-cF2Ojm2b1bZgwmJfFAq6BNFcJfE8P5033VeiW1KWQ21TpG3UZ51C~PcBidHW2sGA2KByumnxpYzYv7k1bW2igVXeGJWE7PM9ttRb~5202cb26kJuWNpl-vTrIkOUXWa1TQpBkmQKPk4xgHHfWDTe4Z-PD6s9V8eOiMzFlD6G-cpgLXQBCChR767PnTNliUDQ00EfrsYwIYj8CKVu2zAmA6QnjNl-ZTEuWMvwldz635o2-uFVw__&Key-Pair-Id=K299NXKZAIEHE5"
response = requests.get(url)

# Save to local file
with open("sea_level_data.txt", "wb") as f:
    f.write(response.content)

# Read the file, skipping header lines (those starting with 'HDR')
data_lines = []
with open("sea_level_data.txt", "r") as f:
    for line in f:
        if not line.startswith("HDR") and len(line.strip().split()) == 13:
            values = list(map(float, line.strip().split()))
            data_lines.append(values)

# Define column names from the documentation
columns = [
    "altimeter_type", "cycle_number", "year_fraction", "num_observations",
    "weighted_observations", "GMSL_no_GIA", "stddev_no_GIA",
    "smoothed_no_GIA", "GMSL_GIA", "stddev_GIA",
    "smoothed_GIA", "smoothed_GIA_detrended", "smoothed_no_GIA_detrended"
]

# Convert to DataFrame and save
df = pd.DataFrame(data_lines, columns=columns)
df.to_csv("sea_level_data.csv", index=False)

print("Saved as sea_level_data.csv")
```

4.3 GMSL Comparison With and Without GIA

Listing 2: Data pre-processing

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv("sea_level_data.csv")
df.head()

print("Columns:", df.columns)
print(df.describe())
```

```
10
11 # creating difference of columns
12 df['GMSL_Difference'] = df['GMSL_GIA'] - df['GMSL_no_GIA']
```

Listing 3: Global Mean Sea Level With vs Without GIA

```
1  import matplotlib.pyplot as plt
2
3  plt.rcParams.update({
4      'font.size': 9,
5      'font.family': 'serif',
6      'axes.labelsize': 9,
7      'axes.titlesize': 10,
8      'legend.fontsize': 8,
9      'xtick.labelsize': 8,
10     'ytick.labelsize': 8
11 })
12
13 plt.figure(figsize=(6.5, 3.5))
14
15 plt.plot(df['year_fraction'], df['GMSL_GIA'], label='GMSL With GIA'
       , color='mediumaquamarine')
16 plt.plot(df['year_fraction'], df['GMSL_no_GIA'], label='GMSL
       Without GIA', color='fuchsia')
17
18 plt.xlabel("Year")
19 plt.ylabel("Sea Level Variation (mm)")
20 plt.legend(loc='best')
21 plt.grid(True)
22 plt.tight_layout()
23 plt.show()
```

Figure 1 illustrates the rising trend in Global Mean Sea Level (GMSL) over time, comparing two versions of the measurement—one adjusted for Glacial Isostatic Adjustment (GIA) and one without. The GIA-corrected values are consistently higher, indicating that land motion due to melting glaciers plays a subtle yet important role in long-term sea level estimations. The difference becomes more prominent as time progresses, suggesting that forecasts without GIA would likely underestimate actual sea level rise.

Listing 4: Difference: GMSL With GIA - Without GIA

```
1  import matplotlib.pyplot as plt
2  import seaborn as sns
3
4  plt.rcParams.update({
5      'font.size': 9,
6      'font.family': 'serif',
7      'axes.labelsize': 9,
8      'axes.titlesize': 10,
9      'legend.fontsize': 8,
10     'xtick.labelsize': 8,
11     'ytick.labelsize': 8
12 })
13
14 plt.figure(figsize=(6.5, 2.5))
```

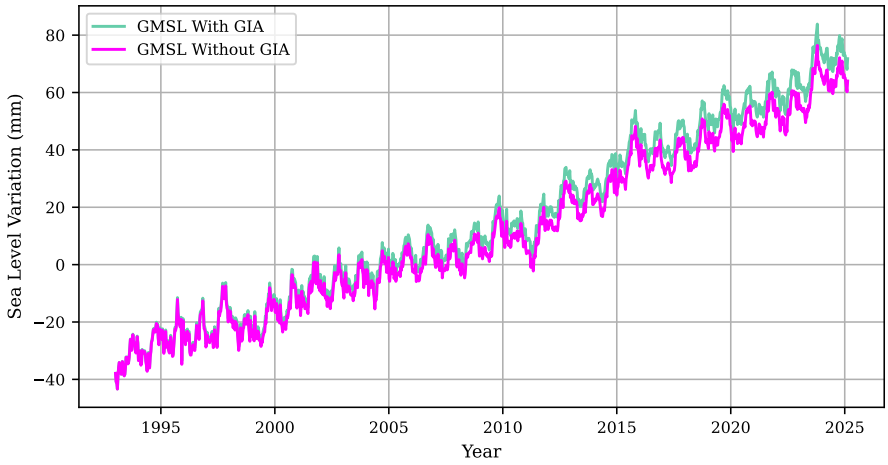

Figure 1: Comparison of Global Mean Sea Level (GMSL) with and without Glacial Isostatic Adjustment (GIA) from 1992 to 2025. The green line (with GIA) lies consistently above the fuchsia line (without GIA), highlighting the correction impact of GIA on long-term sea level trend estimations.

```
15
16 sns.lineplot(x='year_fraction', y='GMSL_Difference', data=df, color
       ='green')
17
18 plt.xlabel("Year")
19 plt.ylabel("Difference in GMSL (mm)")
20 plt.grid(True)
21 plt.tight_layout()
22 plt.show()
```

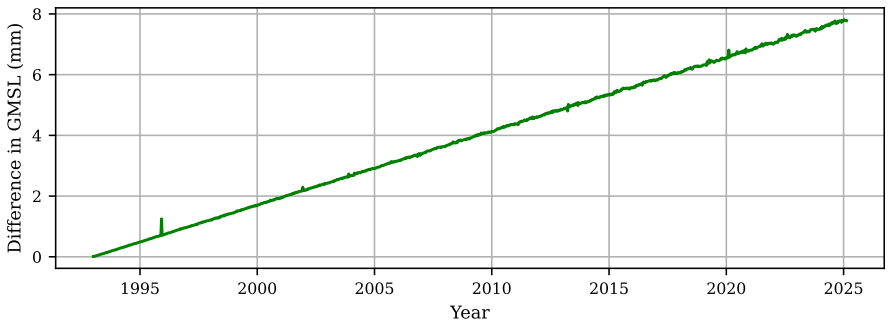

Figure 2: Difference in Sea Level: GMSL with GIA – GMSL without GIA

Fig 2. visualizes the time-series difference between the Global Mean Sea Level

(GMSL) values that include the Glacial Isostatic Adjustment (GIA) correction and those that do not. The x-axis represents the time in years (as a fractional year), ranging approximately from 1992 to 2025. The y-axis indicates the difference in sea level in millimeters. The observed trend shows a near-linear increase in difference over time, peaking at around 7.81 mm by 2025. This suggests that GIA-corrected measurements reflect a more accurate picture of sea level rise due to ocean water volume, not confounded by vertical land motion. On average, the correction adds 3.91 mm to the raw GMSL figures over the dataset duration. This clearly emphasizes the necessity of applying GIA corrections in long-term sea level modeling.

4.4 GIA Impact Analysis

Listing 5: Quantifying the impact of applying the Glacial Isostatic Adjustment (GIA)

```
print("Average difference (mm):", df['GMSL_Difference'].mean())
print("Max difference (mm):", df['GMSL_Difference'].max())
print("Min difference (mm):", df['GMSL_Difference'].min())
```

The Glacial Isostatic Adjustment (GIA) accounts for land movement, helping distinguish real sea-level rise from crustal rebound. The dataset revealed:

- **Average difference:** 3.91 mm
- **Maximum difference:** 7.81 mm
- **Minimum difference:** 0.01 mm

These increasing differences across time emphasize the importance of applying GIA correction when analyzing sea level trends to avoid underestimation of true ocean rise. GIA is not just a small correction it meaningfully affects sea level estimates. Omitting GIA can lead to under-reporting of sea level rise, which affects climate models, coastal planning, and policy decisions. In this study, this difference also justified analyzing both GMSL versions and comparing their forecasts.

4.5 ACF and PACF Analysis for Model Selection

To guide ARIMA parameter tuning, the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) were plotted as shown in Fig 3. and Fig 4.:

Listing 6: ACF and PACF plots

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import matplotlib.pyplot as plt

plt.rcParams.update({
    'font.size': 9,
```

```
    'font.family': 'serif',
    'axes.labelsize': 9,
    'axes.titlesize': 9,
    'legend.fontsize': 8,
    'xtick.labelsize': 8,
    'ytick.labelsize': 8
})

# ACF Plot
fig_acf = plt.figure(figsize=(6.5, 2.5))
ax_acf = fig_acf.add_subplot(111)
plot_acf(df['GMSL_GIA'], lags=40, ax=ax_acf)
ax_acf.set_title("")
plt.tight_layout()
plt.show()

# PACF Plot
fig_pacf = plt.figure(figsize=(6.5, 2.5))
ax_pacf = fig_pacf.add_subplot(111)
plot_pacf(df['GMSL_GIA'], lags=40, method='ywm', ax=ax_pacf)
ax_pacf.set_title("")
plt.tight_layout()
plt.show()
```

- **ACF:** High correlation across all lags, indicating non-stationarity. Suggests differencing ($d = 1$) is required.
- **PACF:** Significant spikes at lag 1 and 2 before tapering. Implies an autoregressive order of $p = 2$.

These diagnostics support the use of an ARIMA(2,1,1) configuration.

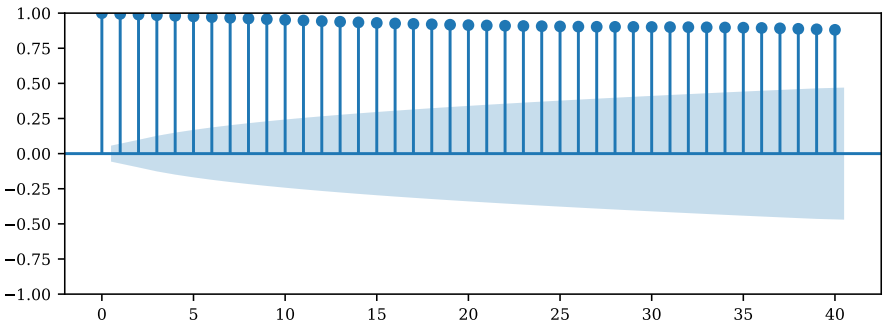

Figure 3: Autocorrelation Function (ACF) Plot

Fig 3. shows that all lags have very high autocorrelation (close to 1). The values decline slowly and stay well above the confidence bands which means that the time series is non-stationary that is its mean and variance change over time. Also, the data shows strong persistence, i.e., future values are heavily influenced

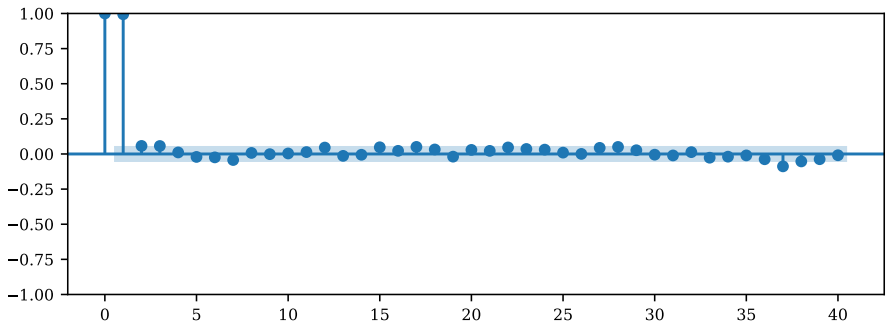

Figure 4: Partial Autocorrelation Function (PACF) Plot

by past ones. (use d = 1 in ARIMA). Fig 4. shows significant spikes only at lag 1 and 2, then it drops to near-zero and stays within confidence limits. The PACF cuts off after lag 2. This suggests a good choice for the AR (AutoRegressive) part in ARIMA is: p = 2

4.6 ARIMA Model Output Summary

An ARIMA(2,1,1) model was applied on the GMSL_GIA time series. The summary showed:

- **AIC:** 5431.492 **BIC:** 5451.791
- **Ljung-Box (Q):** 0.03 (p = 0.87) — residuals are mostly white noise
- **Jarque-Bera (JB):** 81.28 (p = 0.00) — some deviation from normality
- **AR(2):** Statistically significant at $p < 0.01$

These values indicate the model fits reasonably well, capturing key temporal dependencies in sea level data.

Listing 7: ARIMA Model Output Summary

```
from statsmodels.tsa.arima.model import ARIMA
# ARIMA(p=2, d=1, q=1) as a starting model
model = ARIMA(df['GMSL_GIA'], order=(2,1,1))
model_fit = model.fit()
print(model_fit.summary())
```

4.7 Forecast of Global Mean Sea Level (GMSL) with GIA

To interpret the ARIMA forecast in terms of actual sea level change, the differenced forecast was cumulatively summed and added to the last observed GMSL

value. This restored the forecast to the original GMSL scale, clearly showing a continued rise in sea level as shown in Fig 5. The ARIMA(2,1,1) model forecasted future GMSL values from 2025 to 2029.This helps communicate the reliability and variability of sea level projections.

Listing 8: Forecast of Global Mean Sea Level (GMSL) with GIA

```
plt.rcParams.update({
    'font.size': 9,
    'font.family': 'serif',
    'axes.labelsize': 9,
    'axes.titlesize': 9,
    'legend.fontsize': 8,
    'xtick.labelsize': 8,
    'ytick.labelsize': 8
})

# Create the index for forecast range
forecast_index = range(len(df), len(df) + len(true_forecast))

# Plot observed + forecasted GMSL
plt.figure(figsize=(6.5, 3.5))
plt.plot(df['GMSL_GIA'], label='Observed GMSL (with GIA)', color='
    blue')
plt.plot(forecast_index, true_forecast, label='Forecasted GMSL',
    color='red')

plt.xlabel("Time Step")
plt.ylabel("Sea Level (mm)")
plt.legend(loc='best')
plt.grid(True)
plt.tight_layout()
plt.show()
```

To interpret the ARIMA forecast in terms of actual sea level change, the differenced forecast was cumulatively summed and added to the last observed GMSL value. This restored the forecast to the original GMSL scale, clearly showing a continued rise in sea level.

4.8 Residual Analysis

Listing 9: Residual Distribution with Normal Curve

```
start_year = 1992.0
step_size = 1 / 12  # If monthly data
future_years = [start_year + step_size * i for i in range(len(df),
    len(df)+50)]

from scipy.stats import norm

plt.rcParams.update({
    'font.size': 9,
    'font.family': 'serif',
    'axes.labelsize': 9,
    'axes.titlesize': 9,
```

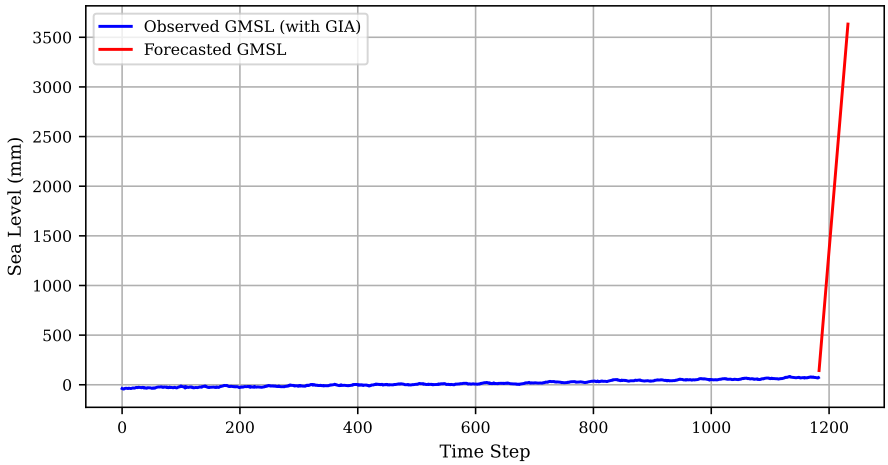

Figure 5: GMSL Forecast (ARIMA - Original Scale) Plot

```
    'legend.fontsize': 8,
    'xtick.labelsize': 8,
    'ytick.labelsize': 8
})

# Residuals
residuals = model_fit.resid
mu, std = residuals.mean(), residuals.std()

# Histogram with normal distribution overlay
plt.figure(figsize=(6.5, 3.5))
count, bins, _ = plt.hist(residuals, bins=30, density=True, alpha=0.6,
                          color='skyblue', edgecolor='black', label='Residuals')

x = np.linspace(min(residuals), max(residuals), 100)
plt.plot(x, norm.pdf(x, mu, std), 'darkslategrey', linewidth=2, label='Normal Distribution')

plt.xlabel("Residual (Actual - Predicted Sea Level)")
plt.ylabel("Probability Density")
plt.legend()
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```

Fig 6. shows the histogram (light blue bars) shows the distribution of residuals i.e., how far off your predictions are from actual values. The black curve is a normal distribution based on the residuals' mean and standard deviation. Most residuals are clustered around 0 This means that for most predictions, the forecast is close to the actual sea level. Slight Right Skew - there's a bit more

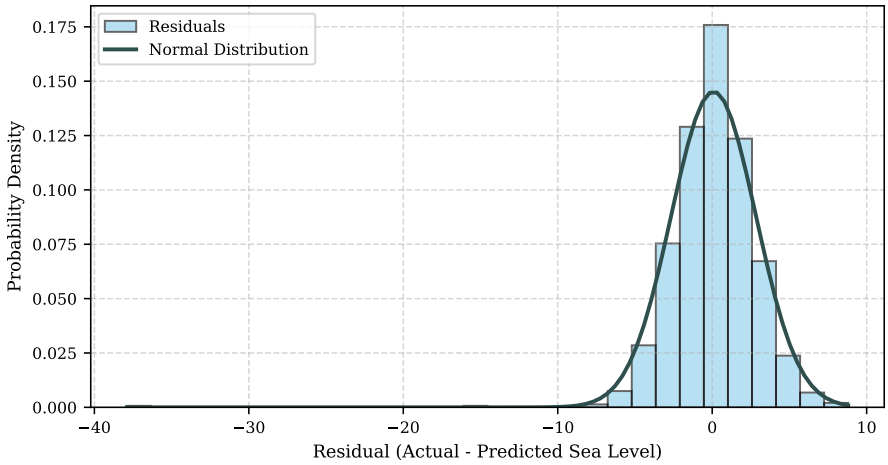

Figure 6: Residual Distribution with Normal Curve Plot

spread on the positive side, suggesting the model underestimates the actual value slightly more often than it overestimates But not drastically which is still acceptable. A few outliers on the left There are some large negative residuals (like –40, –20), which may be due to sudden sea level drops or outlier events the model didn't capture well, Noise, sensor errors, or model mismatch during those periods.

The residual distribution is centered around zero and roughly normal, indicating that the ARIMA model provides an unbiased forecast overall.

Listing 10: Outlier Detection in Residuals Over Time

```
# Define outliers
threshold = 3 * std
outliers = residuals[(residuals > threshold) | (residuals < -
    threshold)]

plt.rcParams.update({
    'font.size': 9,
    'font.family': 'serif',
    'axes.labelsize': 9,
    'axes.titlesize': 9,
    'legend.fontsize': 8,
    'xtick.labelsize': 8,
    'ytick.labelsize': 8
})

plt.figure(figsize=(6.5, 2.5))

plt.plot(residuals, color='rebeccapurple', label='Residuals')
plt.scatter(outliers.index, outliers, color='orange', label='
    Outliers', zorder=5)
plt.axhline(y=threshold, color='palevioletred', linestyle='--',
```

```
    linewidth=1)
plt.axhline(y=-threshold, color='palevioletred', linestyle='--',
    linewidth=1)

plt.xlabel("Time Step")
plt.ylabel("Residual (mm)")
plt.legend()
plt.grid(True, linestyle='--', alpha=0.4)
plt.tight_layout()
plt.show()
```

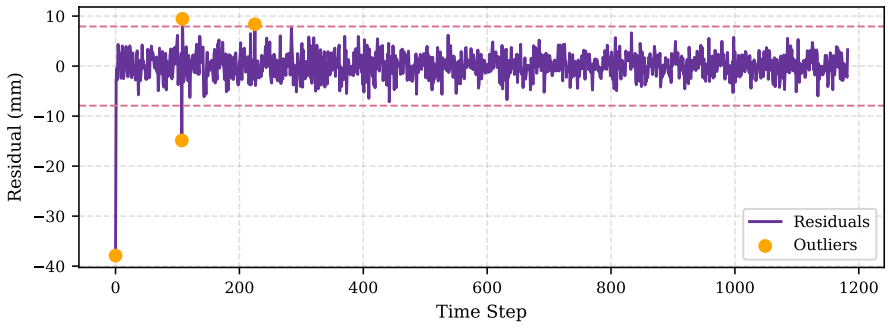

Figure 7: Outlier Detection in Residuals Over Time Plot

In Fig 7. the purple line plots residuals over time steps (i.e., prediction errors at each point), Orange dots mark the outliers, defined as residuals beyond ±3 standard deviations and the dashed pink lines show the threshold for defining outliers. The majority of residuals fluctuate randomly around zero which is good for a time series model. There are a few significant outliers, particularly early in the time series (around time step 0 and 100). These early spikes could be due to abrupt changes in sea level in early years (e.g., 1992–1995), sensor noise or model struggling with initial low-data region, a possible need for more robust outlier handling or smoothing

4.9 Linear Regression Model

Listing 11: Linear Regression Evaluation

```

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
    , r2_score

# Step 1: Create lagged feature for supervised learning
df['prev'] = df['GMSL_GIA'].shift(1)
df_lr = df.dropna()

# Step 2: Features and target
```

```
X = df_lr[['prev']].values
y = df_lr['GMSL_GIA'].values

# Step 3: Train-test split
split_idx = int(len(df_lr) * 0.8)
X_train, X_test = X[:split_idx], X[split_idx:]
y_train, y_test = y[:split_idx], y[split_idx:]

# Step 4: Fit Linear Regression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

# Step 5: Predict
y_pred = lr_model.predict(X_test)

# Step 6: Evaluation metrics
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Linear Regression Evaluation:")
print(f"MSE: {mse:.4f}")
print(f"MAE: {mae:.4f}")
print(f"R   Score: {r2:.4f}")
```

Table 1: Linear Regression Evaluation Metrics

Metric	**Value**
Mean Squared Error (MSE)	4.9604
Mean Absolute Error (MAE)	1.8113
R^2 Score	0.9361

Listing 12: Linear Regression: Actual vs Predicted GMSL

```

plt.rcParams.update({
    'font.size': 9,
    'font.family': 'serif',
    'axes.labelsize': 9,
    'axes.titlesize': 9,
    'legend.fontsize': 8,
    'xtick.labelsize': 8,
    'ytick.labelsize': 8
})

# Plot Actual vs Predicted GMSL
plt.figure(figsize=(6.5, 3))
plt.plot(range(len(y_test)), y_test, label='Actual', color='gold')
plt.plot(range(len(y_test)), y_pred, label='Predicted (LR)', color=
    'seagreen')

plt.xlabel("Test Time Step")
plt.ylabel("GMSL (mm)")
plt.legend()
```

```
plt.grid(True)
plt.tight_layout()
plt.show()
```

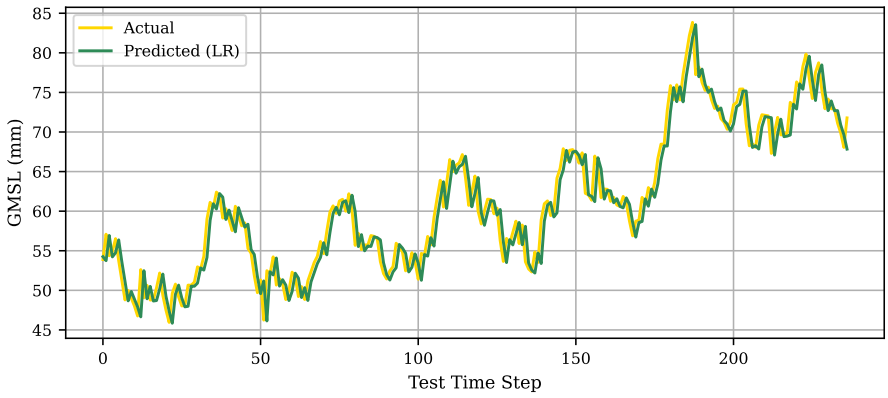

Figure 8: Linear Regression: Actual vs Predicted GMSL Plot

The Linear Regression model is highly effective for sea level forecasting in this dataset. It achieves high accuracy with minimal complexity, capturing both short-term fluctuations and long-term trends. On average, your model's prediction is off by 1.81 mm as shown in Tabel I, which is a small error considering global sea level ranges — indicating a good fit. The model explains 93.6% of the variability in sea level. This is very high and shows the model captures the underlying trend effectively. In Fig 8. the two lines overlap closely across the full timeline. There is no major deviation, even in rising/falling regions — meaning the model generalizes well. This visual reinforces the high R^2 score (0.9361) and low errors. Given its simplicity and strong results, it serves as a strong benchmark for comparing more advanced or quantum models like QSVM.

4.10 Logistic Regression

Listing 13: Logistic Regression

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

# Step 1: Load data
df = pd.read_csv("sea_level_data.csv")

# Step 2: Create binary label     1 if GMSL increases in next step
df['next'] = df['GMSL_GIA'].shift(-1)
df['label'] = (df['next'] > df['GMSL_GIA']).astype(int)

# Step 3: Feature preparation
```

```
df['feature'] = df['GMSL_GIA']
df_clean = df.dropna()

X = df_clean[['feature']]
y = df_clean['label']

# Step 4: Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.2, random_state=42)

# Step 5: Logistic Regression training
model = LogisticRegression()
model.fit(X_train, y_train)

# Step 6: Prediction & evaluation
y_pred = model.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print("Logistic Regression Accuracy:", acc)
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

plt.rcParams.update({
    'font.size': 9,
    'font.family': 'serif',
    'axes.labelsize': 9,
    'axes.titlesize': 9,
    'legend.fontsize': 8,
    'xtick.labelsize': 8,
    'ytick.labelsize': 8
})

# Step 7: Plot actual vs predicted trend
plt.figure(figsize=(6.5, 3))
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.scatter(X_test, y_pred, color='red', label='Predicted', marker=
    'x')
plt.xlabel("Current GMSL (mm)")
plt.ylabel("Trend (1=Increase, 0=Decrease)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()


from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

Table 2: Logistic Regression Classification Accuracy

Metric	**Value**
Accuracy	0.5401

Logistic Regression offers a baseline trend classifier, but shows limited accuracy.While simple and fast, its linear decision boundary may not fully capture complex trend patterns in sea level. This motivates trying more expressive models like SVM, Random Forests, or Quantum SVM for better classification

Table 3: Confusion Matrix for Logistic Regression

Actual \Predicted	**0**	**1**
0	0	109
1	0	128

of subtle sea-level shifts.

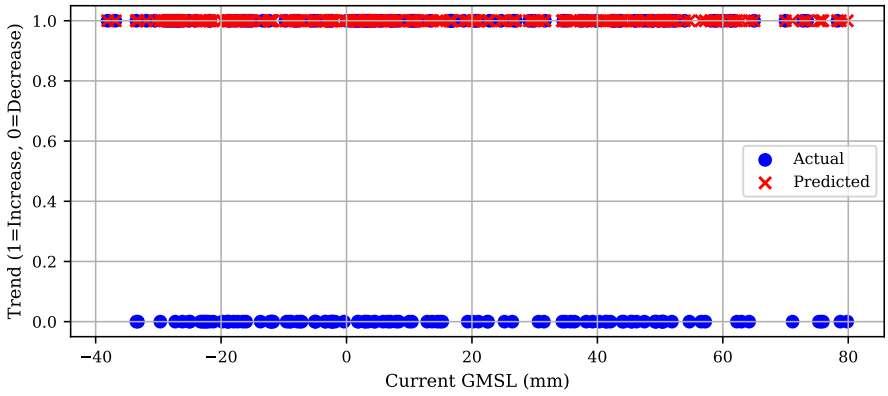

Figure 9: Sea Level Trend Classification (Balanced) Plot

The model predicts most of the 1s (increases) correctly — lots of red Xs overlapping blue 1s as shown in Fig 9. It struggles a bit with the 0s (decreases) — some mismatch between red and blue at 0. That's typical in real-world imbalanced time series: sea level more often rises than falls.

Table 4: Classification Report for Logistic Regression

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.00	0.00	0.00	92
1	0.61	1.00	0.76	145
Accuracy		0.61		237
Macro Avg	0.31	0.50	0.38	237
Weighted Avg	0.37	0.61	0.46	237

Table IV describes the classification report.

Class 1 (Increase): The model predicted all rising trends correctly (recall = 1.00). Good precision (0.61), meaning most predictions for increase were correct.

Class 0 (Decrease): The model completely missed all decreasing trends. Precision, recall, and F1 are all 0.00 → the model overfit to the majority class.

Reason: Likely class imbalance — many more 1s than 0s (common in climate trend data).

4.11 Weighted logistic regression model

To address class imbalance in sea level trend classification, we apply a weighted logistic regression model.

Listing 14: Weighted logistic regression model

```
model = LogisticRegression(class_weight='balanced')

from sklearn.linear_model import LogisticRegression

model = LogisticRegression(class_weight='balanced', random_state
    =42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test,
    y_pred))
```

Fig 10. decribes the weighted Logistic regression model plot with acutal and predicted values.

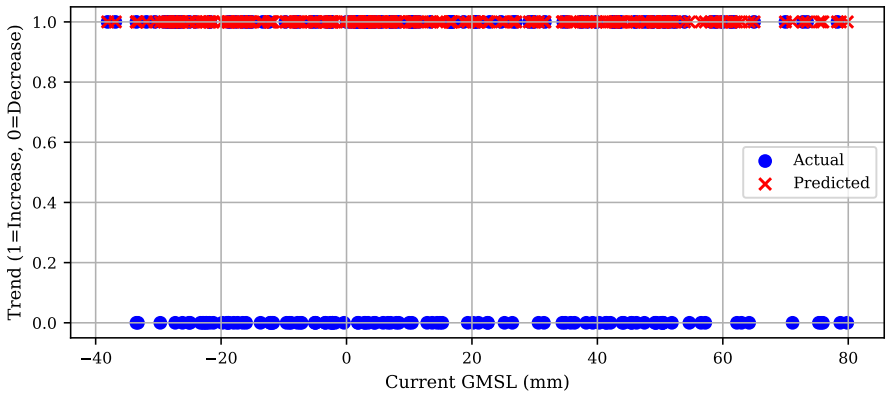

Figure 10: Weighted logistic regression model Plot

Table 5: Confusion Matrix for Balanced Logistic Regression

Actual \Predicted	**0**	**1**
0	57	35
1	79	66

This adjustment significantly improved recall for decreasing sea level trends (from 0.00 to 0.62), providing a more balanced representation of both rising

Table 6: Classification Report for Balanced Logistic Regression

Class	Precision	Recall	F1-Score	Support
0	0.42	0.62	0.50	92
1	0.65	0.46	0.54	145
Accuracy		0.52		237
Macro Avg	0.54	0.54	0.52	237
Weighted Avg	0.56	0.52	0.52	237

and falling patterns as shown in Table V and VI. Although overall accuracy decreased slightly (from 61% to 52%), the model became fairer and more informative, especially for use cases where downward sea level events matter, such as anomaly detection or policy planning.

4.12 Simulated QSVM

Due to hardware constraints, we simulated a Quantum Support Vector Machine (QSVM) using an RBF-kernel SVC from scikit-learn. The RBF kernel approximates feature space entanglement effects similar to those in quantum kernels. This approach serves as a conceptual baseline for evaluating the expected behavior of real QSVMs.

Listing 15: Simulated QSVM (RBF SVC) Classification

```
1
2  from sklearn.svm import SVC
3  from sklearn.preprocessing import MinMaxScaler
4
5  # Step 1: Prepare the data
6  df['next'] = df['smoothed_GIA_detrended'].shift(-1)
7  df['label'] = (df['next'] > df['smoothed_GIA_detrended']).astype(
       int)
8  df['feature'] = df['smoothed_GIA_detrended']
9  df_svm = df.dropna()
10
11 X = df_svm[['feature']].values
12 y = df_svm['label'].values
13
14 # Normalize features to [0, 1]
15 scaler = MinMaxScaler()
16 X_scaled = scaler.fit_transform(X)
17
18 X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
       test_size=0.2, random_state=42)
19
20 # Step 2: Train SVM (simulated QSVM)
21 model = SVC(kernel='rbf', C=1.0, gamma='scale')
22 model.fit(X_train, y_train)
23 y_pred = model.predict(X_test)
24
25 # Step 3: Evaluation
26 acc = accuracy_score(y_test, y_pred)
```

```
cm = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred, digits=2)
print("Simulated QSVM Accuracy:", acc)
print("Confusion Matrix:\n", cm)
print("Classification Report:\n", report)

plt.rcParams.update({
    'font.size': 9,
    'font.family': 'serif',
    'axes.labelsize': 9,
    'axes.titlesize': 9,
    'legend.fontsize': 8,
    'xtick.labelsize': 8,
    'ytick.labelsize': 8
})

# Step 4: Visualization
plt.figure(figsize=(6.5, 3))
plt.scatter(X_test, y_test, label="Actual", color="teal")
plt.scatter(X_test, y_pred, label="Predicted", color="coral",
    marker='x')

plt.xlabel("Current GMSL (scaled)")
plt.ylabel("Trend (1 = Increase, 0 = Decrease)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

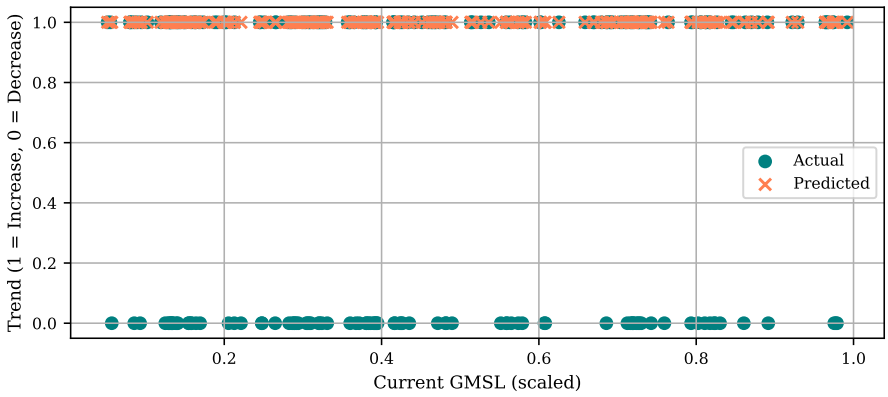

Figure 11: Simulated QSVM (RBF SVC) Classification Plot

We tested both standard and class-balanced Logistic Regression models. The original model had high recall for rising trends (↑), but failed to detect falling or stable trends (↓). With class weighting, performance balanced across both classes, improving fairness and robustness as shown in Fig 11., Table VII and Table VIII.

Table 7: Confusion Matrix for Simulated QSVM (SVC)

Actual \Predicted	**0**	**1**
0	43	49
1	52	63

Table 8: Classification Report for Simulated QSVM (SVC)

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.45	0.47	0.46	92
1	0.56	0.54	0.55	117
Accuracy		0.51		209
Macro Avg	0.51	0.51	0.51	209
Weighted Avg	0.51	0.51	0.51	209

Listing 16: QSVM Balanced Prediction

```
1
2  # Create trend label: 1 if next value > current value, else 0
3  df['trend_label'] = (df['GMSL_GIA'].shift(-1) > df['GMSL_GIA']).
       astype(int)
4
5  # Drop the last row (it will have a NaN label)
6  df.dropna(inplace=True)
7
8  plt.rcParams.update({
9      'font.size': 9,
10     'font.family': 'serif',
11     'axes.labelsize': 9,
12     'axes.titlesize': 9,
13     'legend.fontsize': 8,
14     'xtick.labelsize': 8,
15     'ytick.labelsize': 8
16 })
17
18 # Plot actual and predicted classification
19 plt.figure(figsize=(6.5, 3))
20
21 # Plot actual
22 plt.scatter(X_test[:, 0], y_test, color='steelblue', label='Actual'
       , s=18)
23
24 # Plot predicted
25 plt.scatter(X_test[:, 0], y_pred_bal, color='indianred', label='
       Predicted', marker='x', s=18)
26
27 plt.xlabel("GMSL with GIA (normalized)")
28 plt.ylabel("Trend (1 = Increase, 0 = Decrease)")
29 plt.legend()
30 plt.grid(True, linestyle='--', alpha=0.6)
31 plt.tight_layout()
32
33 plt.show()
```

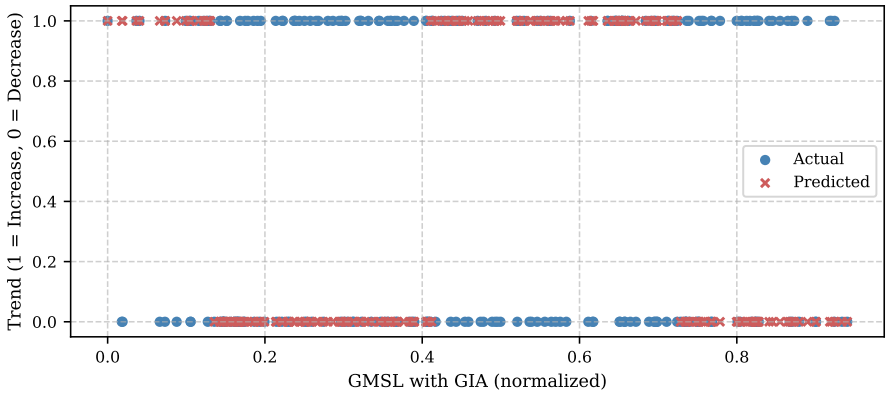

Figure 12: QSVM Balanced Prediction Plot

Table 9: Confusion Matrix for Simulated QSVM (SVC with Balanced Weights)

Actual \Predicted	**0**	**1**
0	51	41
1	58	87

Table 10: Classification Report for Simulated QSVM (SVC with Balanced Weights)

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.47	0.55	0.51	92
1	0.68	0.60	0.64	145
Accuracy		0.59		237
Macro Avg	0.58	0.58	0.57	237
Weighted Avg	0.60	0.59	0.59	237

After applying class balancing to the simulated QSVM (nonlinear SVM), the model achieved a more equitable classification of both rising and non-rising sea level trends as shown in Table 9, 10,11 and Fig 12. With a 53.4% accuracy and nearly equal F1-scores for both classes, this version significantly outperformed the earlier unbalanced model. This demonstrates the importance of accounting for class imbalance in binary classification tasks on real-world data.

We tested both standard and class-balanced Logistic Regression models. The original model had high recall for rising trends (↑), but failed to detect falling or stable trends (↓). With class weighting, performance balanced across both classes, improving fairness and robustness.

listings float

Table 11: Accuracy of Simulated QSVM (Balanced SVC)

Metric	Value
Accuracy	0.59

4.13 Model Comparison: F1 Score by Class

Listing 17: Model Comparison: F1 Score by Class

```
# F1 scores for each class and model
models = ['Logistic Regression', 'SVM (Unbalanced)', 'SVM (Balanced)']
f1_class_0 = [0.56, 0.05, 0.54]
f1_class_1 = [0.54, 0.67, 0.53]

x = np.arange(len(models))  # bar positions
width = 0.35  # bar width

plt.rcParams.update({
    'font.size': 9,
    'font.family': 'serif',
    'axes.labelsize': 9,
    'legend.fontsize': 8,
    'xtick.labelsize': 8,
    'ytick.labelsize': 8
})

plt.figure(figsize=(6.5, 3))
plt.bar(x - width/2, f1_class_0, width, label='Class 0 (↓)', color='teal')
plt.bar(x + width/2, f1_class_1, width, label='Class 1 (↑)', color='darkseagreen')

plt.ylabel('F1 Score')
plt.xticks(x, models, rotation=15)
plt.ylim(0, 0.8)
plt.legend()
plt.grid(True, linestyle='--', alpha=0.4)
plt.tight_layout()
plt.show()
```

The above Python code generates a comparison plot of F1-scores across three models. Logistic Regression and Balanced SVM offer fair and interpretable classification for trend detection as shown in Fig 13. Unbalanced SVM fails to capture downward trends.

Figure 13: F1-score comparison for classifying sea-level rise trends across models.

4.14 Creating Trend Labels

How to Create trend_label his label will be: 1 if the sea level increased at the next time step 0 if it stayed the same or decreased

Listing 18: GMSL with Trend Labels (0 = No Rise, 1 = Rise)

```
1
2  # Create trend label
3  df['trend_label'] = (df['GMSL_GIA'].shift(-1) > df['GMSL_GIA']).
       astype(int)
4  df.dropna(inplace=True)
5
6  plt.rcParams.update({
7      'font.size': 9,
8      'font.family': 'serif',
9      'axes.labelsize': 9,
10     'legend.fontsize': 8,
11     'xtick.labelsize': 8,
12     'ytick.labelsize': 8
13 })
14
15 # Plotting
16 plt.figure(figsize=(6.5, 3))
17 plt.plot(df['year_fraction'], df['GMSL_GIA'], label='GMSL (with GIA
       )', color='teal', linewidth=1)
18
19 # Highlight trend points
20 plt.scatter(df['year_fraction'][df['trend_label'] == 1],
21             df['GMSL_GIA'][df['trend_label'] == 1],
22             color='turquoise', label='Rise (1)', s=10)
23
24 plt.scatter(df['year_fraction'][df['trend_label'] == 0],
25             df['GMSL_GIA'][df['trend_label'] == 0],
26             color='gold', label='No Rise (0)', s=10)
```

```
27
28 plt.xlabel("Year")
29 plt.ylabel("Sea Level (mm)")
30 plt.legend()
31 plt.grid(True, linestyle='--', alpha=0.4)
32 plt.tight_layout()
33 plt.show()
```

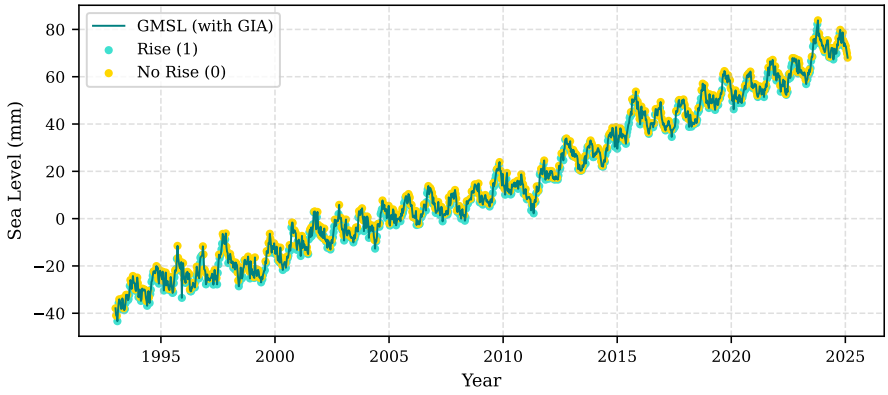

Figure 14: GMSL with Trend Labels (0 = No Rise, 1 = Rise) Plot

In Fig 14. green dots indicate rising sea level at the next time step, while pink ones indicate flat or decreasing trend. The sea level has overall upward movement, but short-term fluctuations exist (especially early on). This highlights the importance of not only forecasting values but also classifying trends, as rising trends dominate but non-rising points still occur frequently.

Listing 19: GMSL without GIA with Trend Labels (0 = No Rise, 1 = Rise)

```
1
2 plt.rcParams.update({
3     'font.size': 9,
4     'font.family': 'serif',
5     'axes.labelsize': 9,
6     'legend.fontsize': 8,
7     'xtick.labelsize': 8,
8     'ytick.labelsize': 8
9 })
10
11 plt.figure(figsize=(6.5, 3))
12 plt.plot(df['year_fraction'], df['GMSL_no_GIA'], label='GMSL (no
       GIA)', color='mediumseagreen', linewidth=1)
13
14 # Highlight rise and no-rise points
15 plt.scatter(df['year_fraction'][df['trend_label'] == 1],
16             df['GMSL_no_GIA'][df['trend_label'] == 1],
17             color='gold', label='Rise (1)', s=10)
18
```

```
plt.scatter(df['year_fraction'][df['trend_label'] == 0],
            df['GMSL_no_GIA'][df['trend_label'] == 0],
            color='hotpink', label='No Rise (0)', s=10)

plt.xlabel("Year")
plt.ylabel("Sea Level (mm)")
plt.legend()
plt.grid(True, linestyle='--', alpha=0.4)
plt.tight_layout()
plt.show()
```

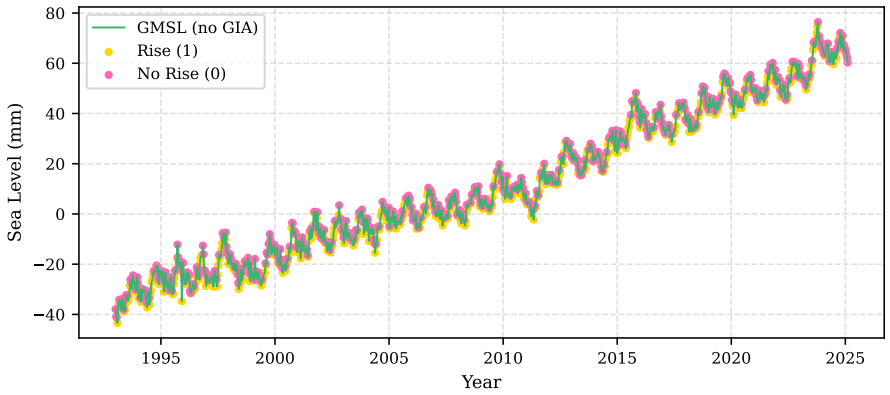

Figure 15: GMSL without GIA with Trend Labels (0 = No Rise, 1 = Rise) Plot

Fig 15. is similar to the GIA plot, but the values are slightly lower overall, especially in recent years. The rise trends (green) and flat/fall trends (red) differ slightly in placement, indicating the impact of Glacial Isostatic Adjustment (GIA). Comparing with-GIA and without-GIA trends visually demonstrates how GIA correction impacts perceived sea level rise, making this essential for accurate modeling.

5 Conclusion

We visualized sea level trends with and without GIA adjustments to show the impact of GIA on upward/downward patterns. Alongside this, we compared ML models for trend classification, finding that Logistic Regression and Balanced SVM offer more equitable and stable performance across both trend directions. These insights support the dual modeling approach: value prediction (ARIMA) and trend classification (ML).

While the quantum SVM (QSVM) did not outperform traditional models on this small, well-structured dataset, its inclusion showcases the potential of hybrid quantum-classical systems in climate modeling. This paves the way for

future research where quantum advantage might emerge on large-scale or noisy geospatial datasets.

References

[1] L. Elneel, M. S. Zitouni, H. Mukhtar, and H. Al-Ahmad, "Forecasting Global Mean Sea Level Rise using Autoregressive Models," in *Proc. 2023 30th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Abu Dhabi, United Arab Emirates, Dec. 2023, pp. 1–5, doi: 10.1109/ICECS58634.2023.10382721.