# Geomagnetic Storm and Flight Delay Correlation Model

*Exploring whether geomagnetic storm activity correlates with commercial flight delays using a machine learning framework.*

## General Information

* **Project Name:** Geomagnetic Storm & Flight Delay Correlation Model
* **Version:** 1.0 (2025-10-11)
* **Description:** This project provides research code to analyze and model the correlation between geomagnetic storm indices and airplane flight delays. It combines historical flight delay data with geomagnetic storm data and uses statistical analysis and machine learning classifiers to determine if geomagnetic disturbances have a measurable impact on flight delay occurrence.

## Project Overview

This repository contains code developed as part of a research study investigating the *possible correlation between geomagnetic storms and flight delays*. The goal is to determine whether indicators of geomagnetic activity (e.g., **Dst**, **Ap**, **Kp** indices) have any significant relationship with delays in commercial airline flights. The project focuses on flights between JFK and LAX in 2023, merging their on-time performance data with space weather data on corresponding dates. The **flight delay** is treated as a binary outcome (0 = *no delay*, 1 = *delayed*), and the code evaluates whether geomagnetic storm metrics can predict or explain these delays better than chance.

**Methodology:** The code performs data cleaning, feature engineering, correlation analysis, and classification modeling: - **Data Integration:** Three datasets (not included in the repo for size/privacy) are required as inputs:  
1. *Flight Delay Data* – A CSV file of flights (JFK–LAX, 2023) including date, time, and total delay minutes for each flight.  
2. *Monthly Geomagnetic Storm Summary* – An Excel file listing counts/durations of disturbed (geomagnetically active) days per month (with columns for number of disturbed days d1, d2, ... and quiet days q1, q2, ...). This is used to derive monthly aggregates like **average\_disturbed** and **average\_quiet** days per month.  
3. *Daily Geomagnetic Indices* – A CSV file of daily geomagnetic readings (e.g., **Dst** index, **Ap** index, **Kp\_max**, **IMF Bz/Bt**, solar wind speed, etc.) for the relevant dates.

These datasets are merged on the date field to create a comprehensive table of flight outcomes alongside geomagnetic metrics. (Any flight or date entries lacking corresponding geomagnetic data are dropped to ensure alignment.)

* **Feature Engineering:** Besides the raw geomagnetic indices from the daily data, the code creates additional features:
* **average\_disturbed / average\_quiet:** computed from the monthly storm data to quantify general geomagnetic activity level in that month.
* **Time of Day Features:** Flight local departure times are converted to minutes since midnight and further transformed into cyclical features (sin\_hour, cos\_hour) to capture potential time-of-day effects (since geomagnetic impacts might differ between day/night, etc.).
* **Threshold Flags:** Binary flags like high\_kp (e.g., Kp\_max ≥ 6), strong\_storm (Dst ≤ -70 nT), bz\_neg (IMF Bz ≤ -10) are generated to mark days with especially strong geomagnetic conditions, and night to mark flights occurring during late-night hours. These engineered features help the models capture non-linear or threshold effects.
* **Correlation Analysis:** The merged dataset is first analyzed statistically to check for any linear or rank correlation between geomagnetic metrics and flight delays. The code computes **Pearson correlation** coefficients (equivalent to point-biserial correlation for a binary target) as well as **Spearman** and **Kendall** rank correlations between each feature and the delay outcome. Results are printed and visualized as a heatmap. In this analysis, the absolute correlation values are examined – if they are very low (close to 0), it indicates little linear relationship. The code also attempts simple transformations (like taking absolute values of Dst, log-transforming Ap, squaring Kp) to see if that improves correlation with the delay variable. Additionally, it filters subsets of data (e.g., only days with strong storms) to see if correlation becomes more pronounced under extreme conditions.
* **Classification Models:** The core of the project is a **machine learning model suite** that treats the problem as a binary classification (delay vs no-delay). By using established classifiers, we can test if geomagnetic data improves predictive accuracy:
* **Logistic Regression Model:** A baseline logistic classifier (from sklearn.linear\_model.LogisticRegression) is used to predict delay occurrence. It is incorporated in a pipeline with feature scaling (StandardScaler) and optionally polynomial feature expansion (sklearn.preprocessing.PolynomialFeatures) to capture interaction effects between geomagnetic features. The code uses a grid search (GridSearchCV) to tune regularization (L1 vs L2, regularization strength C) for a logistic model with second-degree interactions among the storm indices. This logistic model is a simple, interpretable approach to see if any linear or low-order non-linear combination of the indices has predictive power.
* **Random Forest Model:** A non-linear ensemble model (sklearn.ensemble.RandomForestClassifier) is trained to capture more complex relationships. The random forest can implicitly account for interactions and non-linear thresholds in the data. The code uses a balanced class weight (since typically there might be imbalance between no-delay and delay instances) and fits a forest with several hundred trees.
* **(Optional) XGBoost Model:** If **XGBoost** (xgboost.XGBClassifier) is installed, the code will also train a gradient-boosted tree model as another powerful non-linear classifier. (If XGBoost is not available, that step is skipped with a message.)
* All models are trained on a training subset (80% of the data) and then evaluated on a held-out test subset (20%). A 5-fold cross-validation (StratifiedKFold) is also utilized in some cases (e.g., for single-feature screening and during grid search) to ensure results are robust and not dependent on one particular train/test split.
* **Model Evaluation:** For each model, the code outputs standard classification metrics and visualizations:
* After training, a **confusion matrix** is computed (sklearn.metrics.confusion\_matrix) and displayed as a heatmap (using **Seaborn** sns.heatmap) to show true vs predicted outcomes. A **classification report** (sklearn.metrics.classification\_report) is printed, giving precision, recall, and F1-score for the delay vs no-delay classes, as well as overall accuracy.
* **ROC-AUC and PR-AUC:** The models' probabilistic predictions are evaluated with **ROC AUC** (Receiver Operating Characteristic Area Under Curve) and **PR AUC** (Precision-Recall Area Under Curve). The code uses roc\_auc\_score and average\_precision\_score to calculate these metrics on the test set. It also uses RocCurveDisplay.from\_predictions and PrecisionRecallDisplay.from\_predictions to plot the ROC curve and Precision-Recall curve for each model, providing a visual sense of model performance. The ROC curve is especially useful for illustrating the true positive rate vs false positive rate trade-off. An area under the ROC curve of 0.5 means no better than random guessing, whereas 1.0 means perfect separation of classes.
* **Feature Importance:** For the Random Forest (and XGBoost, if run), the code computes **permutation importance** (sklearn.inspection.permutation\_importance) to rank which features have the most influence on the model's predictions. A bar chart is displayed showing the top features and their importance values, helping interpret which geomagnetic indicators (or engineered features like time of day) were most predictive of delays in these models.
* **Single-Feature Screening:** As an additional exploration, the code evaluates each feature in isolation by training a simple logistic model on that single feature (with cross-validation). It prints and plots the cross-validated ROC-AUC for each individual feature. This helps identify if any single geomagnetic index alone has any predictive signal (even if weak).

Overall, this project provides a framework to demonstrate the end-to-end process of data merging, correlation analysis, and model training to answer the question: **"Do geomagnetic storms significantly affect airline delays?"** All scientific interpretations of the results (the actual answer to that question) are discussed in the accompanying research paper, whereas this repository focuses on the code and methodology. *In brief, initial results suggest that any correlation is subtle—standard correlation coefficients are low, and classification models achieve only modest predictive accuracy—implying that geomagnetic effects (if any) on flight timeliness are likely minor compared to other factors.* The logistic model includes a final evaluation that prints a conclusive sentence based on accuracy (e.g., it will output **"True: There is a significant correlation."** or **"False: Still no significant correlation."** depending on whether the model accuracy exceeds a threshold of 0.70). This serves as a simple criterion to judge significance in context.

**Repository Structure:** The key components in this repository are: - **Multi\_Classification\_with\_model\_evaluators.py** – The main script that performs data loading, feature engineering, correlation checks, and runs the suite of classification models (logistic regression with interactions, random forest, etc.) with evaluation plots. This script is well-commented and is intended to be the primary analysis pipeline. - **LegacyModel.py** – An earlier approach focusing on the initial correlation analysis and a basic logistic regression model. It executes similar steps (data prep and a logistic model) but with a simpler structure. This was used as a proof-of-concept and kept for reference. The multi-classification script expands upon this with more robust techniques. - *(No large data files are included in the repository).* The three input datasets mentioned above must be obtained separately and placed in the appropriate location (see **Usage** below). Because these datasets are standardized or proprietary, they are not part of the code repository.

## Installation

To set up the environment for this project, you will need **Python 3.x** (the code was tested with Python 3.9) and the following Python libraries installed:

* **pandas** – for data manipulation (e.g., reading CSV/Excel, merging, cleaning).
* **numpy** – for numerical computations and array handling.
* **seaborn** and **matplotlib** – for generating plots (correlation heatmaps, bar charts, ROC curves, etc.).
* **scikit-learn** – for machine learning modeling and evaluation (providing LogisticRegression, RandomForestClassifier, StandardScaler, train\_test\_split, StratifiedKFold, GridSearchCV, roc\_auc\_score, RocCurveDisplay, ConfusionMatrixDisplay, etc.).
* **SciPy** – (particularly scipy.stats) for statistical correlation functions like Spearman’s rho and Kendall’s tau.
* **xgboost** (optional) – only needed if you wish to run the XGBoost model; the script will handle absence gracefully if not installed.

You can install the required packages using pip:

pip install pandas numpy seaborn matplotlib scikit-learn scipy xgboost

*(If you prefer, a requirements.txt can be created by capturing these dependencies with specific version numbers. Ensure scikit-learn is up-to-date to support the functions used, e.g., version 1.0+ for RocCurveDisplay.)*

**System Requirements:** The code is not very computationally intensive. A standard computer with a few GB of RAM is sufficient, as the dataset (a year of flights and corresponding geomagnetic data) is moderate in size (a few thousand records). The code should run on any OS (Windows, macOS, Linux), but note that file paths in the scripts are currently Windows-style and will need to be adjusted for other systems. No special hardware (like GPU) is required since models are relatively small.

**Before Running:** Ensure you have the three input files available (see **Project Overview** above for descriptions). Because these files are not part of the repository, you will need to provide them yourself: - JFK-LAX flights 2023 Model.csv (flight delay data)  
- Geomagnetic Storms - duration.xlsx (monthly storm data)  
- Geomagnetic-Storm\_Data.csv (daily geomagnetic indices data)

Update the file path variables at the top of the scripts to point to where these files are stored on your system. For example, in the Python files you will see lines like:

flight\_data\_path = r"C:\Users\...\JFK-LAX flights 2023 Model.csv"

These should be edited to the correct path (or a relative path) on your machine. Alternatively, place the data files in a known directory (e.g., a data/ subfolder in this project) and modify the code to use those paths. **If the paths are not set correctly, the code will fail to load the data.**

## Usage

Once the environment is set up and data paths are configured, you can run the analysis. There are two main ways to use the code:

1. **Running as Scripts:** You can execute the Python scripts directly (e.g., via a terminal or command prompt). For a full analysis, run Multi\_Classification\_with\_model\_evaluators.py. For example:

* python Multi\_Classification\_with\_model\_evaluators.py
* This will start the data processing and model evaluations. The script will print progress messages and results to the console. Throughout execution, several windows or interactive plots will appear (correlation heatmaps, ROC curves, etc.). You may need to close or interact with these plot windows for the script to continue if running outside of a notebook. The script is sequential and will pause at each plt.show() call until the plot window is closed. Make sure to allow each plot to render to see the results of each stage.

If you prefer a quicker initial run, you can run LegacyModel.py similarly. It will perform a subset of the analysis (mostly focusing on the logistic regression and correlation metrics). This can be a faster way to get a summary of results, but it is less comprehensive than the main script.

1. **Using a Jupyter Notebook:** For an interactive exploration, you might load the code into a Jupyter Notebook or an interactive Python environment (e.g., IPython or JupyterLab). You can copy the code from the scripts into a notebook and run it step by step, which allows you to inspect data at intermediate steps and render plots inline. This approach is useful for research purposes, as you can modify parameters or add print statements and immediately see the effect. (You could also convert the scripts into notebook format if desired.) Ensure the notebook kernel has the required packages installed and remember to run %matplotlib inline or %matplotlib notebook for inline plots.

After running the script, you should observe the following outputs and results:

* **Console Output:** The terminal/console will display textual results such as:
* Counts of how many records are merged and used (e.g., number of flight records after merging with geomagnetic data).
* Correlation values (Pearson, Spearman, Kendall) of each feature with the delay outcome, printed in descending order of absolute value. For example, you might see lines like Dst vs delay (Pearson): -0.05 indicating a very weak negative correlation between Dst index and delays.
* Model training updates or best parameters (e.g., the best hyperparameters found for the logistic regression with interactions).
* Classification reports for each model, showing precision, recall, F1-score for the "no delay" (0) and "delay" (1) classes, and overall accuracy. For instance, a snippet might look like:
* === RandomForest ===   
   precision recall f1-score support   
   0 (no-delay) 0.60 0.70 0.65 100   
   1 (delay) 0.55 0.45 0.49 80   
   ...  
  ROC-AUC: 0.58 | PR-AUC: 0.50
* (The above numbers are just an illustration; actual results depend on the data. The ROC-AUC and PR-AUC values will indicate performance – e.g., 0.58 suggests the model is only slightly better than random guessing in this hypothetical example.)
* **Plots and Figures:** The code will generate several plots to help visualize the findings:

*Example Confusion Matrix:* After a model is evaluated, a confusion matrix heatmap is shown to summarize prediction outcomes. In the example above, the matrix displays the number of flights correctly vs incorrectly classified by the model (darker blue indicates higher counts). The **diagonal cells** (top-left for no-delay, bottom-right for delay) are the correctly predicted cases, while off-diagonals are misclassifications (e.g., top-right cell are flights that were predicted *delay* but actually had *no delay*). This helps in understanding whether the model has a bias towards predicting delays or non-delays. A well-performing model would have most values on the diagonal. In our context, since delays are relatively rare events, the classifier might tend to predict "no delay" often – resulting in a stronger top-left value. The confusion matrix gives a concrete sense of how many delays are being missed versus correctly identified.

*Example ROC Curve:* The Receiver Operating Characteristic curve illustrates the trade-off between true positive rate (TPR) and false positive rate (FPR) for the classifier across different threshold settings. The curve above is an example output from a classifier, with the **ROC AUC** (area under the curve) shown in the legend (e.g., AUC = 0.82 for this illustrative model). The dashed diagonal line represents random performance (AUC 0.5). Points towards the upper left corner indicate better performance (higher TPR and lower FPR). In our geomagnetic correlation model, an ROC curve is generated for each classifier (Logistic Regression, Random Forest, etc.) on the test data. These curves allow us to visually assess how well the model can distinguish delay vs no-delay flights. For instance, if the AUC is around 0.6, the curve will lie only slightly above the diagonal, suggesting only a mild ability to discriminate delays from normal flights. The code also plots a Precision-Recall curve (not shown here) which is useful when dealing with imbalanced classes (since flight delays might be less frequent, the PR curve focuses on how precise and complete the delay predictions are). Generally, the model results in this project showed relatively low AUC values, reinforcing that geomagnetic indices alone have limited predictive power for flight delays.

* *Correlation Heatmaps:* Early in the analysis, the script will show heatmaps of the correlation matrix. One heatmap highlights the correlation of each feature with the delay outcome (with values typically very low, often under 0.1 in absolute terms). Another heatmap might show the full Pearson correlation matrix among all features and delay. These visuals help confirm that no single geomagnetic parameter has a strong linear relationship with flight delays (the cells are mostly light-colored, indicating correlation near 0).
* *Feature Importance Bar Charts:* After training the Random Forest or XGBoost, a bar chart of the top features by importance is displayed. This can reveal which factors the model considered most useful. For example, the **time of day** features or flags like night might rank high if flight delays have more to do with departure time than geomagnetic conditions. If a geomagnetic index like Kp\_max appears in the top features, it suggests some influence (though not necessarily causal). In our trials, we might see that no geomagnetic feature dominates the importance chart, aligning with the idea that their contribution is small.
* *Single-Feature ROC bars:* A horizontal bar plot shows the ROC-AUC for classifiers built on each feature alone. This is essentially a univariate screening – a bar reaching ~0.5 means that feature by itself has no predictive skill, whereas higher bars (closer to 0.6-0.7) would indicate some predictive signal. This plot is useful for identifying if any single metric stands out. For instance, one might discover that **Dst index** alone yields a slightly higher AUC than others (hypothetically, say 0.58 vs ~0.52 for others), but still nothing very strong.

**Interpreting Results:** The expectation going into this analysis is that geomagnetic storms have at best a subtle effect on flight operations (perhaps through affecting communications or radiation levels, etc.). After running the model, you may find that the classification accuracy and AUC values are not much above random chance. This would mean the model is not very effective at predicting delays from geomagnetic data alone – implying little to no direct correlation. On the other hand, if you find a model or specific metric that does improve predictive power appreciably, that would be evidence in favor of a correlation. The provided code infrastructure makes it easy to experiment with different features or thresholds to further investigate this. For more nuanced discussion on the scientific implications, refer to the research publication associated with this project.

## License

This project has *no explicit license specified at this time*. All rights are reserved to the author. If you wish to reuse or adapt the code, please contact the author to discuss permissions. (It is recommended to add an open-source license in the future to clarify usage rights.)

## Contact Information

**Author/Maintainer:** Arsh (GitHub username: **GoodDeveloper2018**).

For any questions, feedback, or collaboration inquiries, you can reach out by creating an issue on this GitHub repository or contacting the author directly through their GitHub profile. Since this is research code, constructive feedback and replication results are welcome. If you use this code or data in your own work, please credit the author and reference the associated research paper (once published).