City Predictor: Leveraging Machine Learning for Urban Insights

Colton Blackwell

1. Problem Statement

A company aims to build a new development and needs to identify the most suitable city that meets specific criteria such as high population, low elevation, and other relevant parameters. The objective of this project is to develop and utilize a machine-learning predictive model that can analyze various city attributes and recommend the best city globally for the company's project based on the provided criteria.

1.1 Quick Use Case Overview

The company defines/adjusts the desired parameters (e.g., population, elevation, location features) and inputs these features into the model. The machine-learning model predicts the geographic coordinates (latitude and longitude) that best match the specified criteria. Subsequently, the model displays a 3D figure of Earth, plotting the predicted point and identifying the nearest city that best fits the company's development needs. The figure also shows the distance (in km) between the predicted point and the identified best-fit city.

2. Data Preprocessing

The dataset used for this CMPT 353 project was chosen from the following source: Geonames dataset. This dataset was gathered using a simple .read_csv() function.

The pre-processing tasks included eliminating rows where the features exceeded the maximum allowable values, transforming the 'Coordinates' attributes into separate 'latitude' and 'longitude' columns to facilitate easier model predictions, and ensuring all numerical values were indeed numeric. Unnecessary features, such as administrative codes and modification dates, were removed, and rows with missing coordinate values were excluded. Empty population and elevation values were filled with -1, and rows with nonsensical longitude and latitude values (e.g., latitude > 90, longitude > 180) were removed.

Outliers were identified using methods such as graphing and human perception, Z-score, IQR, and box plots. After completing the pre-processing stage, the dataset was saved into processed_cities_dataset.csv, which was then used to train and evaluate the machine learning model.

3. Classification Model

A Random Forest regressor model is a powerful tool in machine learning, particularly useful for predictive tasks where the relationship between input features and output targets is complex or nonlinear. It leverages the ensemble learning technique by constructing multiple decision trees during training and outputs the average prediction of those trees for regression tasks. This model excels in handling large datasets with high-dimensional features and is robust against overfitting, a common issue in single decision tree models. By averaging predictions from multiple trees, Random Forests reduce variance and improve generalization, making them effective in scenarios with noisy data or where feature importance is unclear. Additionally, they are relatively insensitive to outliers, offering flexibility in data types and distributions.

The processed dataset was split into 80% for training and 20% for testing. Since the Random Forest Regressor model cannot handle non-numerical data, categorical features were transformed into numerical codes using the `Categorical.codes()` method. As detailed in section 3.1 below, the chosen parameters were employed for model evaluation in section 3.2.

3.1 Hyperparameter Tuning

Using RandomizedSearchCV() on our Random Forest Regression model, it was determined that the following are approximately the best parameters:

Hyperparameters	Latitude	Longitude
max_depth	20	None
max_features	None	‘log2’
min_samples_leaf	1	1
min_samples_split	10	2
n_estimators	86	159

3.2 Measuring Model Performance

The model performance for predicting latitude and longitude is excellent. High R^2 and Explained Variance Scores indicate that both models fit the data well, while low Median Absolute Errors show that predictions are accurate. Overall, these results suggest that the models are highly reliable, with the longitude model performing slightly better.

	Latitude	**Longitude**
R^2 Score	0.9859	0.9964
Median Absolute Error (MAE)	1.3965	1.6894
Explained Variance Score	0.9890	0.9965
Median Absolute Error	0.7652	1.0253

4. Results/Findings/Conclusions

Based on our analysis in section 3.2, we can confidently conclude that the RandomForestRegressor model is a reliable choice for this dataset. The model demonstrated excellent performance in predicting city locations, attributed to its high accuracy, robustness to overfitting, and versatility. Notably, the model's accuracy improved with additional training data, as evidenced by the reduced distance between predicted and actual coordinates.

Utilizing hyperparameter tuning methods like RandomSearchCV() helped mitigate overfitting, a common issue with tree-based models. The ensemble effect of the random forest also contributed significantly. While individual decision trees exhibit low bias but high variance, the Random Forest reduces variance through bagging and the ensemble effect without significantly increasing bias, resulting in a more balanced model that generalizes better to unseen data.

In conclusion, we identified a problem that required a solution, prepared our data meticulously for valid model training, and optimized our datasets. We then constructed and fine-tuned our regression model using advanced techniques like hyperparameter tuning. Our well-tuned model successfully predicted outcomes on the test dataset, demonstrating the effectiveness of our approach in creating a robust regression solution.

5. Lessons Learned and Future Work

Spending time on data cleaning and preprocessing is crucial as it ensures the model isn't learning from noisy or incorrect data. Handling missing values, outliers, and encoding categorical variables correctly helps in building a more robust and accurate model. Handling outliers can be improved further by exploring advanced outlier detection algorithms such as Isolation Forest to identify and handle outliers more effectively.

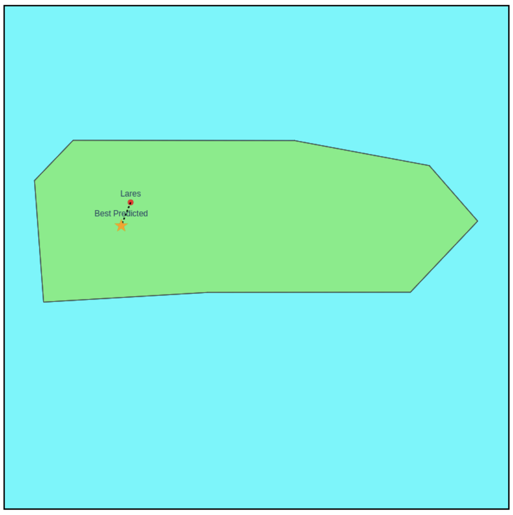
We encountered several challenges, including accurately identifying features that were unnecessary for predicting coordinates. Other issues included incorrect data formatting, missing values, and managing the bias-variance tradeoff to prevent overfitting and underfitting.

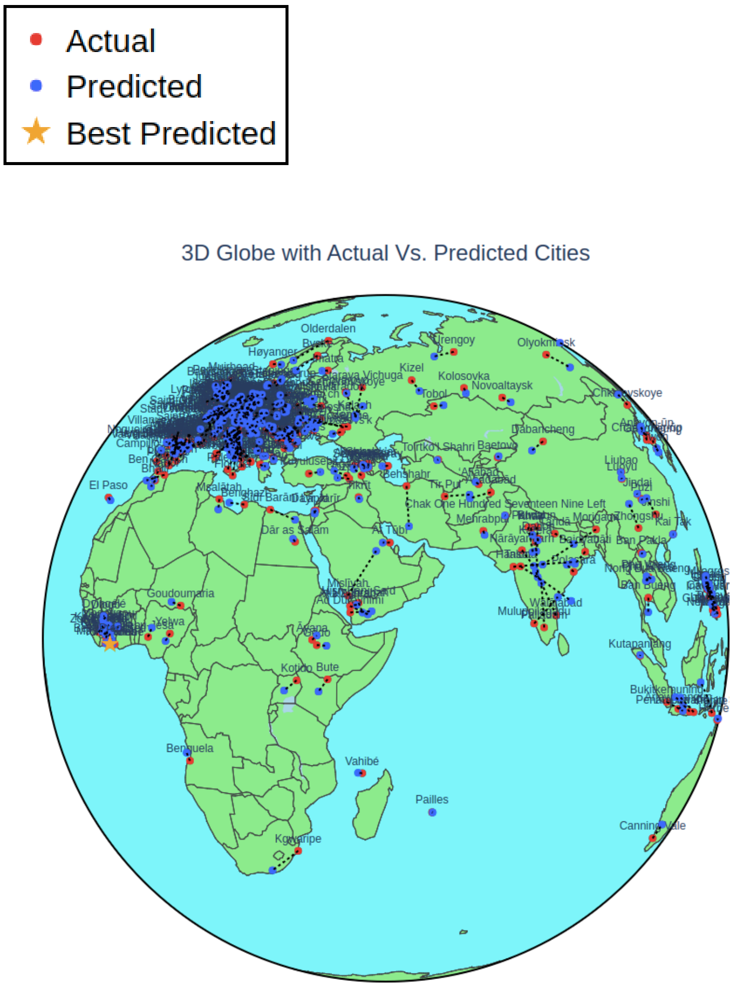
If we had more time, we could have implemented additional improvements, such as conducting more extensive hyperparameter tuning using techniques like GridSearchCV to find the optimal parameters for our model. Additionally, we could have utilized advanced feature selection methods, such as recursive feature elimination, to identify and remove irrelevant features more effectively.

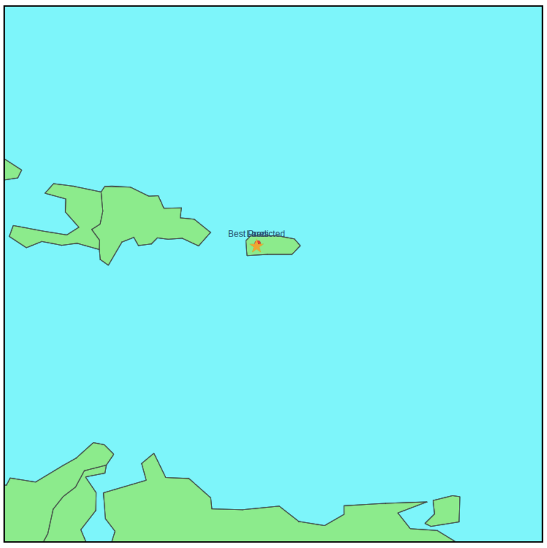
6. Gallery/Demo

****<u>Please</u> View Video Demo (1m:18s)!**

Link: https://www.youtube.com/watch?v=SAnjm8y8Jd0







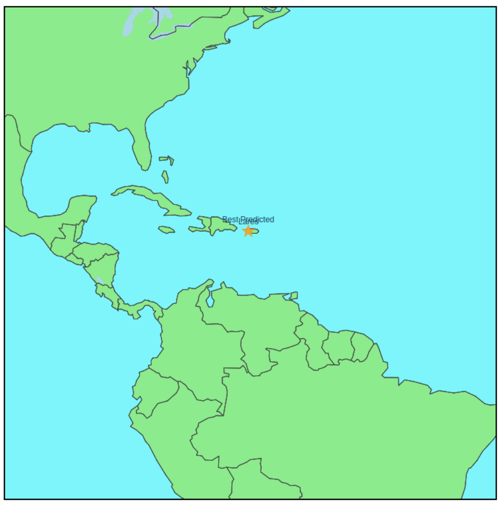


Fig. 1 (top right): 3D Globe with Actual Vs. Predicted Cities where n_cities = 500.

Fig. 2- 4 (Left): 3D Globe with Actual Vs. Predicted Cities where n_cities = 1. Each image moving down the page is Zooming outward. This use case resembles How a company would use this program To locate the best city based on their specified Parameter values (Ex. Population = 50k, Elevation = 300, etc.)

Project Experience Summary

Accomplish Statements

Colton Blackwell

- Pre-processed the .csv dataframe for efficient model training and evaluation. This included identifying and eliminating unnecessary features (using the .drop() command) and transforming categorical attributes into codes. This transformation was accomplished using the Categorical.codes() method, as the machine-learning model requires numerical values as inputs. After completing additional pre-processing steps (handling missing values and outliers), the resulting dataset was efficient and ideal for a regression prediction task.

- Discovered ideal hyperparameters for the machine-learning model using the RandomizedSearchCV() hyperparameter-tuning method. This involved defining a range of valid parameter values and fitting the model 40 times to find the best combination of values. The best resulting values are shown in Section 3.1. This task was necessary to balance overfitting and underfitting by adjusting parameters such as max_depth and min_samples_split for the best fit.

- Utilized the Random Forest Regressor model to predict coordinates based on parameters such as population, elevation, and location features. Cross-validation was used to split the processed dataset into training and validation sets. The model was then trained using the .fit() method and used to predict locations with the .predict() method. Understanding the context helped identify that this regression model would likely perform better compared to others.

- Evaluated model performance using metrics such as mean_absolute_error() and r2_score() to justify the use of the RandomForestRegressor ML model. Achieved impressive results, as shown in Section 3.2, proving that the RandomForestRegressor with the hyperparameter values obtained in Section 3.1 is a valid model for this task.

- Identified outliers in the dataset that could disrupt model predictions. This involved using Z-score functions and box-plot visuals to identify such points, and then removing any points that did not make sense. Observable outliers were eliminated if they did not fit the context.

- Visualized predicted vs. actual points on a 3D globe. This allowed us to observe the accuracy of our model. This step also offered a fun and interactive method to measure the success of our project.