QUESTIONS

1. What dataset did you use? How many samples? Labeled? Unlabeled? Features?

a) What type of data, and what is your data about?

- Combination of textual (categorical) and numerical (continuous). Used 23221 samples out of the Global dataset of 94511 samples, by filtering out Asian Time Zone.

b) If it is labeled, how many samples do you have in minority and majority class of the data?

- The dataset is labeled; each observation contains a temperature_celsius value serving as the target variable for supervised regression.
- The target variable temperature_celsius was defined early to streamline subsequent preprocessing, ensure consistent feature&label separation, and maintain clarity across all modeling steps.
- As this is a regression task, the target variable temperature_celsius is continuous and therefore does not have majority or minority classes. Instead, the target distribution is slightly right-skewed, with most observations between ~20°C (21.3) and ~35°C (32.2), mean ≈ 25.9°C, and range from -25°C (29.9) to 49°C (49.2).

c) Provide a brief description of the features, and a link to the data.

- Total Features: 41
    - 30 numerical (continuous) and 10 categorical variables.
    - Target(Label): 1 temperature_celcius

- Major Feature Categories:
    - Meteorological Variables: humidity, pressure (mb), wind (kph), precipitation (mm), visibility (km), UV index
    - Air-Quality Metrics: CO, O_3, NO_2, SO_2, PM2.5, PM10, EPA/DEFRA indices
    - Astronomical Features: sunrise, sunset, moon phase, illumination
    - Geolocation / Time: latitude, longitude, timezone, last_updated
    - Feature List
    - country: Country of the weather data
    - location_name: Name of the location (city)
    - latitude: Latitude coordinate of the location
    - longitude: Longitude coordinate of the location
    - timezone: Timezone of the location
    - last_updated_epoch: Unix timestamp of the last data update
    - last_updated: Local time of the last data update
    - temperature_celsius: Temperature in degrees Celsius
    - temperature_fahrenheit: Temperature in degrees Fahrenheit
    - condition_text: Weather condition description
    - wind_mph: Wind speed in miles per hour
    - wind_kph: Wind speed in kilometers per hour
    - wind_degree: Wind direction in degrees
    - wind_direction: Wind direction as a 16-point compass
    - pressure_mb: Pressure in millibars
    - pressure_in: Pressure in inches
    - precip_mm: Precipitation amount in millimeters
    - precip_in: Precipitation amount in inches
    - humidity: Humidity as a percentage
    - cloud: Cloud cover as a percentage

- feels_like_celsius: Feels-like temperature in Celsius
- feels_like_fahrenheit: Feels-like temperature in Fahrenheit
- visibility_km: Visibility in kilometers
- visibility_miles: Visibility in miles
- uv_index: UV Index
- gust_mph: Wind gust in miles per hour
- gust_kph: Wind gust in kilometers per hour
- air_quality_Carbon_Monoxide: Air quality measurement: Carbon Monoxide
- air_quality_Ozone: Air quality measurement: Ozone
- air_quality_Nitrogen_dioxide: Air quality measurement: Nitrogen Dioxide
- air_quality_Sulphur_dioxide: Air quality measurement: Sulphur Dioxide
- air_quality_PM2.5: Air quality measurement: PM2.5
- air_quality_PM10: Air quality measurement: PM10
- air_quality_us-epa-index: Air quality measurement: US EPA Index
- air_quality_gb-defra-index: Air quality measurement: GB DEFRA Index
- sunrise: Local time of sunrise
- sunset: Local time of sunset
- moonrise: Local time of moonrise
- moonset: Local time of moonset
- moon_phase: Current moon phase
- moon_illumination: Moon illumination percentage

- Dataset Link
  World Weather Repository (Daily Updating) – Kaggle

d) Analyze your data, any categorical, ordinal, and etc?

i. If you use images, what type of image and what size are they and etc

- The dataset contains heterogeneous features representing multiple data types commonly encountered in real-world tabular datasets:
  - Categorical Variables (10): Textual or nominal attributes used for grouping or encoding.
  - Includes:
    - country, location_name, timezone, condition_text, wind_direction, sunrise, sunset, moon_phase, moonrise, moonset.
  - Numerical Variables (30): Continuous physical measurements with meaningful magnitudes and units.
  - Includes:
    - temperature, humidity, pressure, wind speed, visibility, all air_quality_* indicators
  - Ordinal / Integer-Coded Variables: Represent values on an ordered scale but treated as continuous during modeling.
  - Includes:
    - uv_index, air_quality_us-epa-index, air_quality_gb-defra-index, moon_illumination
  - Target Variable:
    - temperature_celsius - continuous numeric value used as the regression label
  - Image Data:
    - None - The dataset consists entirely of structured tabular data (text + numeric)

2. Use data pre-processing? EDA? Feature engineering.

a) In each category, what techniques did you use?

- **Data Cleaning**
  - Duplicate values: detected and dropped all the duplicate values and validated if any exist after dropping. (0 found)
  - Missing values/Negative values in non-negative features: Identified missing values, replaced negative values with NaN and then imputed numeric features with median, categorical with mode, validated Missing values existence.
- **Outlier Handling**
  - Used IQR method for outlier detection and caped extremes beyond 1.5XIQR for continuous features.
  - Validated capping with Box Plot.

- **EDA (Exploratory Data Analysis)**
  - Histogram of target (temperature_celsius) for distribution shape (in starting as well as after capping)
  - Boxplots (before/after capping) to visualize outliers
  - Correlation heatmap + VIF for multicollinearity assessment

- **Scaling & Transformation**
  - PowerTransformer (Yeo-Johnson) on numerical features.

- **Encoding**
  - One-Hot Encoding of categorical variables (handle_unknown="ignore", min_frequency=0.01)

- **Feature Reduction**
  - Dropped redundant columns (e.g., unit duplicates, time strings, astronomical weak features)

- **Feature Engineering**
  - **Filter:** Removed features with high correlation (VIF > 10 or redundant unit)
  - **Embedding:** Applied PCA (on Power-Transformed numeric features) - examined explained variance (95 % by 11 components)
  - **Wrapper:** Used RFECV with Ridge Regressor (CV=3) - selected 63 optimal features (CV RMSE ≈ 1.89)
  - Plotted graphs for both PCA and RFECV

b) Discuss briefly what did you use

- Data were systematically cleaned and standardized to ensure that all meteorological and air-quality variables were comparable in scale.
- While handling outliers a conundrum on how to handle outliers for temperature_celsius was faced where one choice was to keep the standard code that handles all outliers the same which resulted in removal of most negative temperature values, this was dealt by choice 2 where negative values for temperature will be let to be to maintain real world scenario but then uit would make the model too data specific and result to manual changes every time the dataset changes and the type of data we have. Hence, the choice was reverted to the original choice of keeping it standard so that the code is future proof. The overall result remained the same, though the observation was risen RMSE scores for RF.
- The PowerTransformer effectively corrected positive skewness in variables such as precip_mm, CO, NO_2, $PM_{2.5}$, and PM_{10}, yielding near-Gaussian distributions. (Also suggested by professor over discussion of improvement in results)
- Categorical features were one-hot encoded with frequency-based grouping to avoid high-dimensionality from rare labels.

- The correlation heatmap and VIF analysis confirmed multicollinearity among duplicate unit columns, which were dropped before modeling. Dropping those columns significantly improved VIF and reduced multicollinearity. We have also chose models accordingly in the further steps.
- For dimensionality analysis, Principal Component Analysis (PCA) was applied on the transformed numeric subset, showing that approximately 95 % of the variance could be explained by the first 11 principal components.
- Finally, a wrapper-based feature selection using RFECV (Ridge Regressor, CV = 3) identified 63 optimal encoded features, confirming that model performance stabilized beyond this subset and validating the overall feature-reduction strategy.

c) Please be sure for data pre-processing, address all possible approaches.

- The workflow covered every major preprocessing stage:
- Data validation: detect & replace invalid values
- Imputation: median/mode strategy
- Normalization & scaling: Yeo-Johnson Power Transform
- Outlier treatment: IQR capping
- Encoding: frequency-aware one-hot encoding
- Feature selection: filter → embedding → wrapper
- All transformations were encapsulated inside a ColumnTransformer + Pipeline, ensuring reproducibility and eliminating data leakage during model training.

d) What feature engineering did you use? Use 1 filtering, 1 embedding and 1 wrapping. What do you learn from this?

- Power transformation substantially improved model stability and lowered RMSE.
- Feature selection via RFECV confirmed that redundant correlated attributes could be removed with no loss in accuracy, improving training efficiency.

i. Can you analyze multi-collinearity results?

- High VIF values (> 10) were observed for redundant pairs (°C vs °F, mph vs kph, in vs mm), confirming strong collinearity. After feature reduction, the Variance Inflation Factor (VIF) values dropped significantly across all remaining predictors, confirming improved numerical stability. Although minor residual multicollinearity remained, it was within acceptable limits and did not affect model performance or interpretability.

ii. How many features you started your training process? Removing any features?

- Initial features: 41 total columns in the raw dataset
- After preprocessing & feature reduction: retained 21 columns in the final use_df before encoding
- After one-hot encoding: categorical expansion resulted in ≈ 63 encoded numerical features used for modeling
- After RFECV wrapper selection: optimal feature subset confirmed at 63 encoded features with minimal RMSE (≈ 1.89)

- The final model training was performed using 21 cleaned base columns (≈ 63 encoded variables) after systematically removing redundant, weakly correlated, and non-generalizable attributes, these where saved in use_df (final cleaned dataset before modeling).

3. Training and testing process,

a) From categories of Classification supervised learning, Regression supervised learning or unsupervised learning, select three models,

- The models selected by me where Ridge, Random Forest, and Gradient Boosting Regressor for Regression supervised learning.
- These models were chosen to balance bias, compare linear vs. ensemble behaviors, and examine feature interaction in continuous temperature prediction.

b) Develop these three models, using proper parameters

- All the three models where developed using unified pipeline that incorporated preprocessing.
- The training was done in a 80/20 split with stratification based on temperature quantities to preserve label distribution.
- The parameters used were: -
  - alpha
  - n_estimator
  - max_depth
  - min_sample_leaf
  - max_features
  - learning_rate
- GridSearchCV was applied to each model to find the optimal parameters set using RMSE (Root Mean Squared)as the scoring metric.

c) What does these parameters mean? What values you set?

- Each Parameter has its own duty in the machine larning model, those are as follows:-
  - **alpha –** L2 regularization strength, contols overfitting
  - **n_estimators –** number of trees in ensemble, optimal 100-200
  - **max_depth –** maximum depth of trees, limit model complexity
  - **min_sample_leaf –** minimum samples per leaf node, prevent overfitting in smaller split
  - **max_features –** number of features considered per split
  - **learning_rate –** shrinks contribution of each tree; trade off bias vs variance, optimal ~0.1, give balance between accuracy and generalization

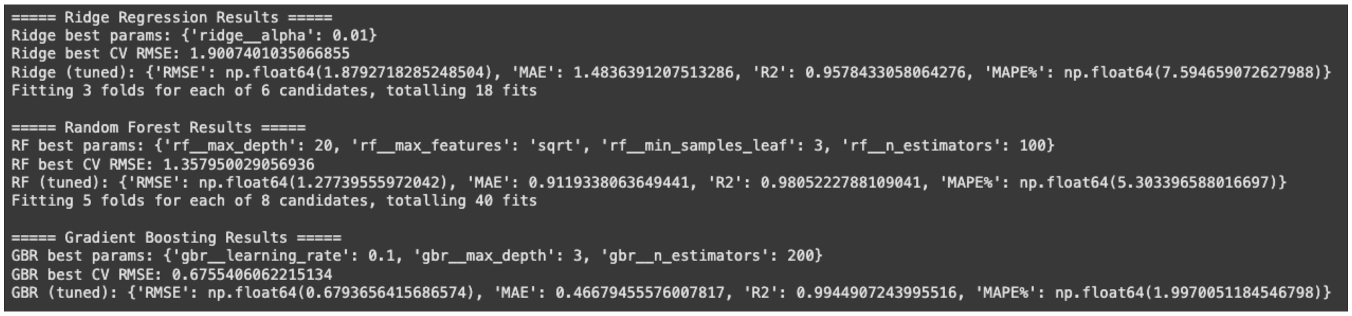
d) Any prediction?

- The model was trained and there where prediction generated on test data of 20% split
- Scatter plot of the predicted vs actual tempretures for all the three regressors showed tight clustering around the indentity line showing model correctness.
- The best performing amodel among all was Gradient Boosting Regressor, as it achieved the lowest RMSE of 0.68 ºC and the highest R^2 of 0.994 making it most reliable for temperature prediction.

4. Select proper hyper parameter tuning technique, and what is the best parameters for models,

i. Define the technique you used

- The tuning process was conducted using Grid Search Cross Validation (GridSearchCV), this technique exhaustively evaluates combinations of model parameters based on a scoring metric.
- The validation strategy was 5 Fold Cross Validation, though in models like Random forest only 3 folds were used to optimize runtime.
- Negative Root Mean Squared Error (neg_root_mean_squared_error) is used as the scoring metrics to sort out the best model.
- The Model is trained on k-1 folds and validated on the remaining folds, repeated k-times.
- The Best Parameters and results after tuning are as follows:-

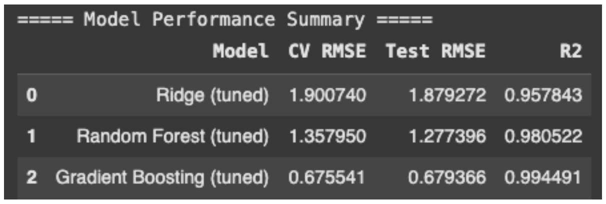


```
===== Ridge Regression Results =====
Ridge best params: {'ridge__alpha': 0.01}
Ridge best CV RMSE: 1.9007401035066855
Ridge (tuned): {'RMSE': np.float64(1.8792718285248504), 'MAE': 1.4836391207513286, 'R2': 0.9578433058064276, 'MAPE%': np.float64(7.594659072627988)}
Fitting 3 folds for each of 6 candidates, totalling 18 fits

===== Random Forest Results =====
RF best params: {'rf__max_depth': 20, 'rf__max_features': 'sqrt', 'rf__min_samples_leaf': 3, 'rf__n_estimators': 100}
RF best CV RMSE: 1.357950029056936
RF (tuned): {'RMSE': np.float64(1.27739555972042), 'MAE': 0.9119338063649441, 'R2': 0.9805222788109041, 'MAPE%': np.float64(5.303396588016697)}
Fitting 5 folds for each of 8 candidates, totalling 40 fits

===== Gradient Boosting Results =====
GBR best params: {'gbr__learning_rate': 0.1, 'gbr__max_depth': 3, 'gbr__n_estimators': 200}
GBR best CV RMSE: 0.6755406062215134
GBR (tuned): {'RMSE': np.float64(0.6793656415686574), 'MAE': 0.46679455576007817, 'R2': 0.9944907243995516, 'MAPE%': np.float64(1.9970051184546798)}
```

- The Observation After Tuning was as follows:-



===== Model Performance Summary =====

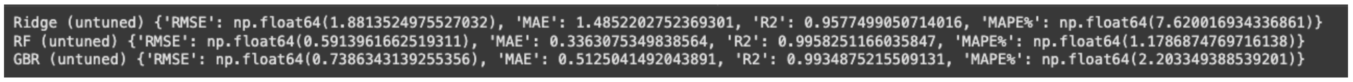
	Model	CV RMSE	Test RMSE	R2
0	Ridge (tuned)	1.900740	1.879272	0.957843
1	Random Forest (tuned)	1.357950	1.277396	0.980522
2	Gradient Boosting (tuned)	0.675541	0.679366	0.994491

5. What metrics you used?

- The metrics used to evaluate the model performance where as follows: -
  - RMSE (Root Mean Squared Error) – measures prediction deviation in ºC
  - MAE (Mean Absolute Error) – calculates the average absolute difference between predicted and actual values
  - R2 (coefficient of Determination) – explaines variance captured by the model
  - MAPE (Mean Absolute Percentage Error) – expresses the MAE as a percentage of the actual values

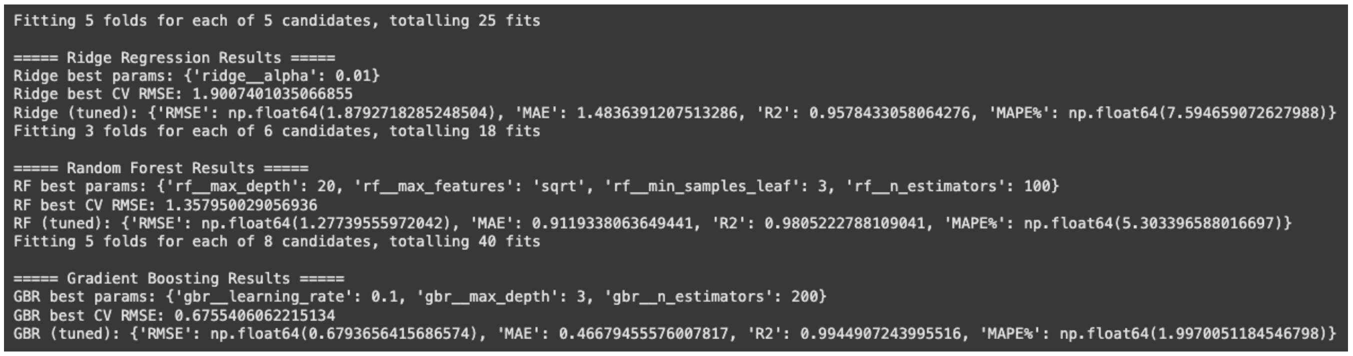
a) Compare the performance of your three models in terms of these metrics,

- Untuned Model Performance



```
Ridge (untuned) {'RMSE': np.float64(1.8813524975527032), 'MAE': 1.4852202752369301, 'R2': 0.9577499050714016, 'MAPE%': np.float64(7.620016934336861)}
RF (untuned) {'RMSE': np.float64(0.5913961662519311), 'MAE': 0.3363075349838564, 'R2': 0.9958251166035847, 'MAPE%': np.float64(1.1786874769716138)}
GBR (untuned) {'RMSE': np.float64(0.7386343139255356), 'MAE': 0.5125041492043891, 'R2': 0.9934875215509131, 'MAPE%': np.float64(2.203349388539201)}
```

- Tuned Model Performance



```
Fitting 5 folds for each of 5 candidates, totalling 25 fits

===== Ridge Regression Results =====
Ridge best params: {'ridge__alpha': 0.01}
Ridge best CV RMSE: 1.9007401035066855
Ridge (tuned): {'RMSE': np.float64(1.8792718285248504), 'MAE': 1.4836391207513286, 'R2': 0.9578433058064276, 'MAPE%': np.float64(7.594659072627988)}
Fitting 3 folds for each of 6 candidates, totalling 18 fits

===== Random Forest Results =====
RF best params: {'rf__max_depth': 20, 'rf__max_features': 'sqrt', 'rf__min_samples_leaf': 3, 'rf__n_estimators': 100}
RF best CV RMSE: 1.357950029056936
RF (tuned): {'RMSE': np.float64(1.27739555972042), 'MAE': 0.9119338063649441, 'R2': 0.9805222788109041, 'MAPE%': np.float64(5.303396588016697)}
Fitting 5 folds for each of 8 candidates, totalling 40 fits

===== Gradient Boosting Results =====
GBR best params: {'gbr__learning_rate': 0.1, 'gbr__max_depth': 3, 'gbr__n_estimators': 200}
GBR best CV RMSE: 0.6755406062215134
GBR (tuned): {'RMSE': np.float64(0.6793656415686574), 'MAE': 0.46679455576007817, 'R2': 0.9944907243995516, 'MAPE%': np.float64(1.9970051184546798)}
```

- The performances of each model according to these metrics are as follows: -
  - **Ridge Regression:** the performance of ridge regression remained almost unchanged after tuning with a minor change in RMSE of ~0.002 decrement after tuning showing stability on normalized data, regularization effectively controlled multicollinearity as well.
  - **Random Forest Regression:** showed signs of slight overfitting dure to high R^2. After tuning with more depth in tree and leaf constraints, though the model traded a bit on accuracy for generalization. Hence, the model became less variance prone.
  - **Gradient Boosting Regressor:** This model delivered the best balance between bias and variance, Tuning significantly reduce RMSE from 0.738 to 0.679 and MAPE from 2.20 to 1.99. With moderate learning rate (0.1) and 200 estimators it achieved the lowest test error and most stable performance.

b) Did you deal with any overfitting/underfitting issues
If yes, how did you solve it?

- Overfitting was dealt with using regularization + controlled tree depth and Power Transformer which contributed to reducing RMSE.
- Along with all this reduction of multicollinearity by reducing the number of colinear features with a very high VIF score also contributed to lesser noise.

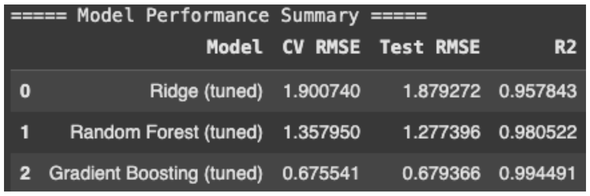
c) Did you use any regularization approach?

- To deal with regularization each model had its own forte.
- Ridge model: Explicit L2 regularization via the alpha parameter (best=0.01)
- Tree-Based Models: implicit regularization using limited depth i.e. max_depth and minimum leaf size i.e. min_samples_leaf.

- Feature preprocessing: I used PowerTransformer instead of StandardScaler which stabilized variance across skewed features and indirectly contributed to reducing overfitting.

d) Get the results with cross validation and without cross validation for each model, any difference? Discuss the method did you use?

- Cross validation confirmed that models were neither data-specific nor overfit. It ensured consistent evaluation before deployment. The difference in results were negligible.



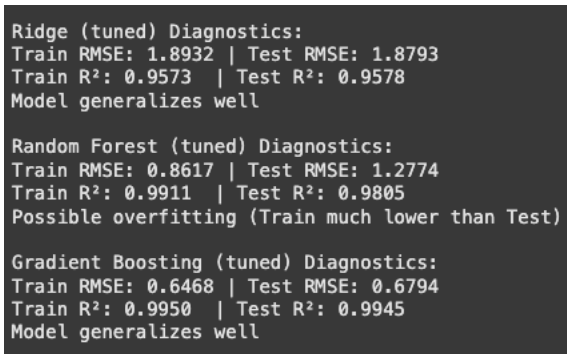
===== Model Performance Summary =====

	Model	CV RMSE	Test RMSE	R2
0	Ridge (tuned)	1.900740	1.879272	0.957843
1	Random Forest (tuned)	1.357950	1.277396	0.980522
2	Gradient Boosting (tuned)	0.675541	0.679366	0.994491

- In Ridge model, there is stable linear generalization, in random forest there is slight variance as seen in the RMSE but that is acceptable, and in Gradient Boosting there is fine-tuned learning rate and depth with enhanced precision.

e) Provide the results without using hyperparameter and with hyper parameter, and compare them

- There is visible improvement in Gradient Boosting Regressor.



```
Ridge (tuned) Diagnostics:
Train RMSE: 1.8932 | Test RMSE: 1.8793
Train R²: 0.9573  | Test R²: 0.9578
Model generalizes well

Random Forest (tuned) Diagnostics:
Train RMSE: 0.8617 | Test RMSE: 1.2774
Train R²: 0.9911  | Test R²: 0.9805
Possible overfitting (Train much lower than Test)

Gradient Boosting (tuned) Diagnostics:
Train RMSE: 0.6468 | Test RMSE: 0.6794
Train R²: 0.9950  | Test R²: 0.9945
Model generalizes well
```

- Ridge has marginal improvement, regularization stabilized the output and Random Forest has slightly higher RMSE but variance was reduced.

6. Analyze the results,

a) What did you learn?

- Throughout this project, I learnt that rather than model training and testing the most important part of prediction or analysis projects using Machine learning in pre-processing. It is very important to put thought behind choosing the right features and working through systematic preprocessing, feature Engineering, and model tuning are in building high-performance regression models for real-world data such as weather.
- I also learnt that each model behaved differently with respect to data normalization, variance and complexity control:
  - Ridge Regression: showed how regularization improved numerical stability in the presence of correlated meteorological variables.
  - Random Forest and Gradient Boosting demonstrated the value of ensemble methods in capturing nonlinear temperature patterns and weather interactions.
- Using PowerTransformer() significantly reduces skewness, enabling tree based and linear models to learn smoother relationships.
- In essence, I learned how to move from Raw Data → Feature Refinement → Model Tuning → Performance Validation, step by step ensuring to make right choices in terms of feature selection, methods and scaling/transforming strategies and analyzing results through the way to build up on each decision I take along the way to tune a model that is not only accurate but also reliable.

b) Did you find any pattern in your models/data?

- Upon initial analysis of data and correlations between them I found that there were many features that had very strong correlations with each other which gave high VIF (Variance Inflation Factor) scores.
- Temperature (^{O}C) is closely related to feels_like_celsius (r ~0.98), humidity (negative correltion), air_pressure (inverse relationship)
- There was also moderate influences from features like wind speed, visibility etc
- Weak relationships with air quality features like (NO2, CO, SO2) contributing minimally.
- Upon close Cross validation and analysis it was observed that Gradient Boosting showed a monotonic rising fit along the y=x line in scatter plot, it had the clusters nearest to the Identity line, showing high accuracy and confirming strong alignment between predicted and actual temperature.

c) Provide few key points that you learned from your results.

- **Normalization and Feature Selection:** Applying PowerTransformer and removing redundant units (°F, mph/kph, etc.) substantially improved consistency across features.
- **Regularization Works:** Ridge Regression's low variance validated the use of small alpha (0.01) for stable learning.
- **Tree Ensembles Excel:** Random Forest captured nonlinearities but slightly overfit; Gradient Boosting achieved optimal bias-variance balance.
- **Feature Reduction Validated:** RFECV confirmed that about 63 encoded features (from 41 original variables) explained most variance; removing the rest avoided noise.
- **Cross-Validation Reliability:** Minimal RMSE difference between CV and test sets showed excellent generalization of tuned models.

d) Any conclusion

- The analysis successfully predicted daily near-surface temperature (°C) for Asian cities using a rich mix of meteorological and air-quality data.
- Among all models, Gradient Boosting Regressor achieved the best predictive accuracy (RMSE = 0.68 °C, R^2 = 0.994), showing superior learning of complex feature relationships.
- Ridge Regression served as a reliable baseline; Random Forest offered interpretability with minor overfitting that was later controlled by hyperparameter tuning.

- The overall experiment validated that data normalization, feature selection, and systematic tuning are indispensable in high-accuracy regression pipelines.
- Future improvements could involve integrating temporal dependencies via recurrent or hybrid models (e.g., LSTM + GBR) for multi-day forecasting.