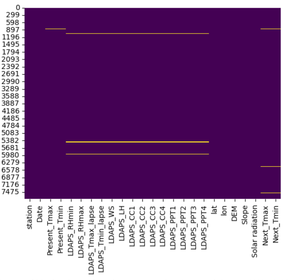
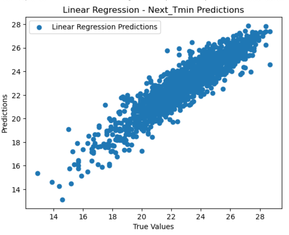
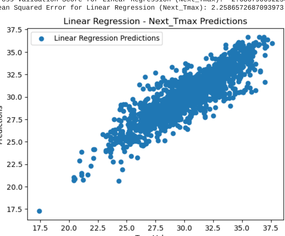
In [1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import HistGradientBoostingRegressor
from sklearn.impute import SimpleImputer
from sklearn.metrics import mean_squared_error

# Load the dataset
link = "https://github.com/dsrscientist/Dataset2/raw/main/temperature.csv"
data = pd.read_csv(link)

# Check for missing values
sns.heatmap(data.isnull(), cbar=False, cmap='viridis')
plt.show()

# Preprocessing and Feature Engineering
# Drop unnecessary columns
data = data.drop(['Date'], axis=1)

# Handle missing values using SimpleImputer
imp = SimpleImputer(strategy='mean')
df_imputed = pd.DataFrame(imp.fit_transform(data), columns=data.columns)

# Separate features and target variables
X = df_imputed.drop(['Next_Tmax', 'Next_Tmin'], axis=1)
y_min = df_imputed['Next_Tmin']
y_max = df_imputed['Next_Tmax']

# Split the data into training and testing sets
X_train, X_test, y_train_min, y_test_min = train_test_split(X, y_min, test_size=0.2, random_state=42)
_, _, y_train_max, y_test_max = train_test_split(X, y_max, test_size=0.2, random_state=42)

# Build models
models = {
    'Linear Regression': LinearRegression(),
    'HistGradientBoosting': HistGradientBoostingRegressor()
}

# Evaluate models
for name, model in models.items():
    # Cross-validation for Next_Tmin
    cv_score_min = cross_val_score(model, X, y_min, cv=5, scoring='neg_mean_squared_error')
    print(f'Cross-Validation Score for {name} (Next_Tmin): {np.mean(cv_score_min)}')

    # Train and test the model for Next_Tmin
    model.fit(X_train, y_train_min)
    pred_min = model.predict(X_test)
    mse_min = mean_squared_error(y_test_min, pred_min)
    print(f'Mean Squared Error for {name} (Next_Tmin): {mse_min}')

    # Visualize predictions for Next_Tmin
    plt.scatter(y_test_min, pred_min, label=f'{name} Predictions')
    plt.xlabel('True Values')
    plt.ylabel('Predictions')
    plt.title(f'{name} - Next_Tmin Predictions')
    plt.legend()
    plt.show()

    # Cross-validation for Next_Tmax
    cv_score_max = cross_val_score(model, X, y_max, cv=5, scoring='neg_mean_squared_error')
    print(f'Cross-Validation Score for {name} (Next_Tmax): {np.mean(cv_score_max)}')

    # Train and test the model for Next_Tmax
    model.fit(X_train, y_train_max)
    pred_max = model.predict(X_test)
    mse_max = mean_squared_error(y_test_max, pred_max)
    print(f'Mean Squared Error for {name} (Next_Tmax): {mse_max}')

    # Visualize predictions for Next_Tmax
    plt.scatter(y_test_max, pred_max, label=f'{name} Predictions')
    plt.xlabel('True Values')
    plt.ylabel('Predictions')
    plt.title(f'{name} - Next_Tmax Predictions')
    plt.legend()
    plt.show()

# Hyperparameter tuning
param_grid = {
    'learning_rate': [0.01, 0.1, 0.2],
    'max_iter': [100, 150, 200],
    'max_depth': [None, 10, 20],
    'min_samples_leaf': [1, 2, 4]
    # Add other parameters based on your model
}

alternative_model = HistGradientBoostingRegressor()
grid_search_min = GridSearchCV(alternative_model, param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search_min.fit(X, y_min)
print(f'Best Parameters for HistGradientBoostingRegressor (Next_Tmin): {grid_search_min.best_params_}')

grid_search_max = GridSearchCV(alternative_model, param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search_max.fit(X, y_max)
print(f'Best Parameters for HistGradientBoostingRegressor (Next_Tmax): {grid_search_max.best_params_}')

# Save the best models
final_model_min = grid_search_min.best_estimator_
final_model_max = grid_search_max.best_estimator_

# Save the models using joblib or pickle
import joblib
joblib.dump(final_model_min, 'best_model_next_tmin.joblib')
joblib.dump(final_model_max, 'best_model_next_tmax.joblib')
```



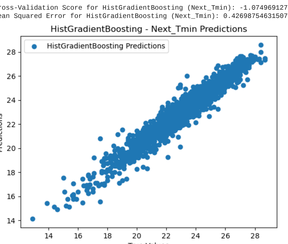

```
Cross-Validation Score for Linear Regression (Next_Tmin): -1.1229851058663127
Mean Squared Error for Linear Regression (Next_Tmin): 0.9644785082807341
```



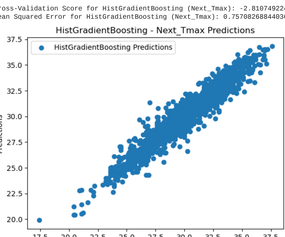

```
Cross-Validation Score for Linear Regression (Next_Tmax): -2.6867906922543385
Mean Squared Error for Linear Regression (Next_Tmax): 2.2586572687093973
```




```
Cross-Validation Score for HistGradientBoosting (Next_Tmin): -1.0749691270772189
Mean Squared Error for HistGradientBoosting (Next_Tmin): 0.4269875463507724
```




```
Cross-Validation Score for HistGradientBoosting (Next_Tmax): -2.810749224392359
Mean Squared Error for HistGradientBoosting (Next_Tmax): 0.7570826884403622
```




```
Best Parameters for HistGradientBoostingRegressor (Next_Tmin): {'learning_rate': 0.1, 'max_depth': None, 'max_iter': 100, 'min_samples_leaf': 1}
Best Parameters for HistGradientBoostingRegressor (Next_Tmax): {'learning_rate': 0.1, 'max_depth': None, 'max_iter': 100, 'min_samples_leaf': 2}
['best_model_next_tmax.joblib']
```

Out[1]:

Importing Libraries:

Imported necessary libraries such as pandas, numpy, seaborn, matplotlib.pyplot, scikit-learn's modules (train_test_split, cross_val_score, GridSearchCV), and specific models (LinearRegression, HistGradientBoostingRegressor)

Loading Dataset:

Loaded the dataset from a given link using pandas. Checking for Missing Values:

Used seaborn's heatmap to visualize and check for missing values in the dataset. Preprocessing and Feature Engineering:

Dropped the 'Date' column as it was considered unnecessary. Handled missing values using SimpleImputer with the mean strategy. Separated features (X) and target variables (y_min and y_max) for both 'Next_Tmin' and 'Next_Tmax'.

Splitting Data:

Split the data into training and testing sets for both 'Next_Tmin' and 'Next_Tmax'. Building Models:

Created instances of Linear Regression and HistGradientBoostingRegressor models. Evaluating Models:

Utilized cross-validation scores and mean squared errors to evaluate and compare model performance for both 'Next_Tmin' and 'Next_Tmax'. Visualized the predictions against true values using scatter plots. Hyperparameter Tuning:

Defined a parameter grid for hyperparameter tuning. Used GridSearchCV to find the best hyperparameters for the HistGradientBoostingRegressor model for both 'Next_Tmin' and 'Next_Tmax'. Saving the Best Models:

Saved the best models using joblib for both 'Next_Tmin' and 'Next_Tmax'. Explanation for Model Selection:

Mentioned that various criteria such as cross-validation scores, mean squared errors, and domain knowledge can be considered for selecting the final models. Documentation:

Encouraged the user to write down their findings, insights, and the steps taken in the Jupyter Notebook. Optional Visualization/Summary:

Suggested creating optional visualizations or tables to summarize the results.