Final Project
Weather Prediction

Karunya Karthikeyan

12/11/2024

Introduction

Problem Statement:

Accurate weather prediction is crucial for various sectors, including agriculture, transportation, and disaster management. The goal of weather prediction is to provide information people and organizations can use to reduce weather-related losses and enhance societal benefits, including protection of life and property, public health and safety, and support for economic prosperity and quality of life *(Council)*.

Motivation and Challenges:

Accurate forecasts help by providing us with early warnings of temperatures high or low, precipitation, snow rate, and similar natural conditions. This can be very helpful for farmers who use them for agricultural management; this can help them plan accordingly to save their crops from the cold, heat or even rain (*Ritchie and Roser*). Weather predictions can also be used to plan transportation accordingly, for instance, the operations of aeroplanes largely depend on the weather conditions. If it is raining too heavily, then flights will be delayed; by predicting the weather, we will be able to make plans accordingly. This will also help us see what global warming will look like in the future on a larger scale.

Approach:

This project aims to develop both a Linear Regression and a Ridge Regression model based on historical meteorological data to predict weather conditions for the next day, based on data collected from previous days. The Linear regression is straightforward while the Ridge Regression uses regularization to handle overfitting, making it suitable for complex datasets. We can see a difference in performance between the two. Then, I also use KNN to classify data into weather conditions, hot or cold, based on the maximum and minimum temperatures.

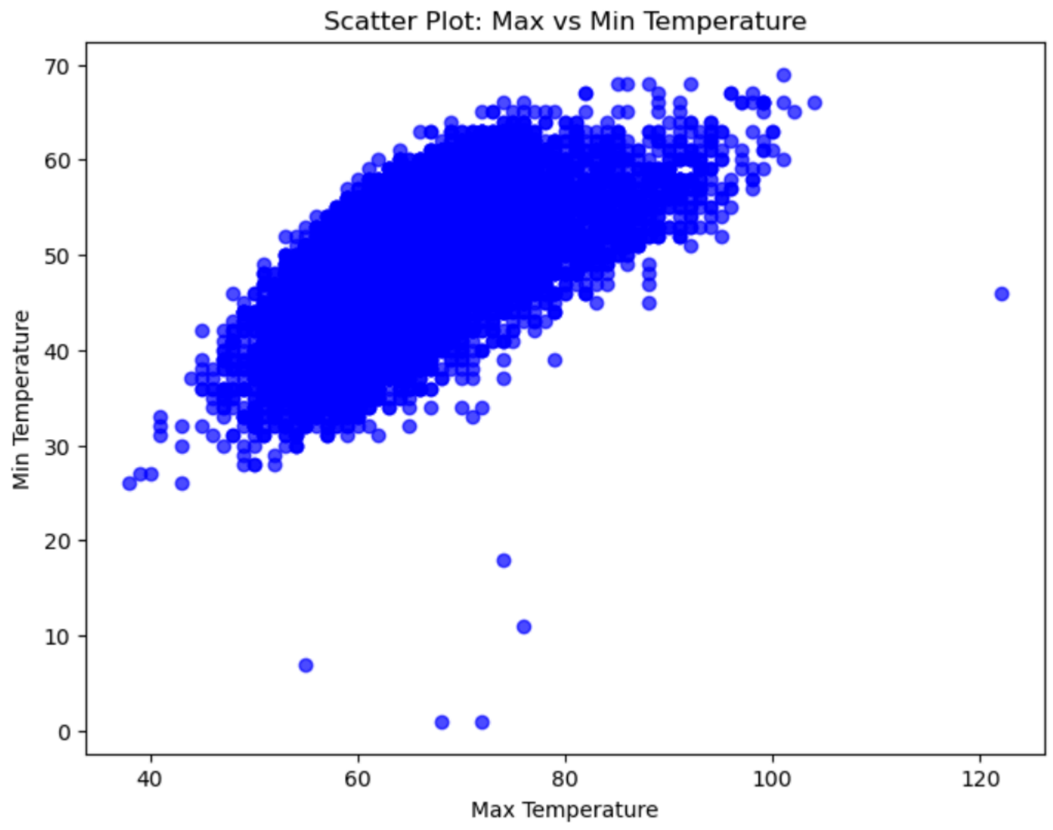
Data/Environment

Data:

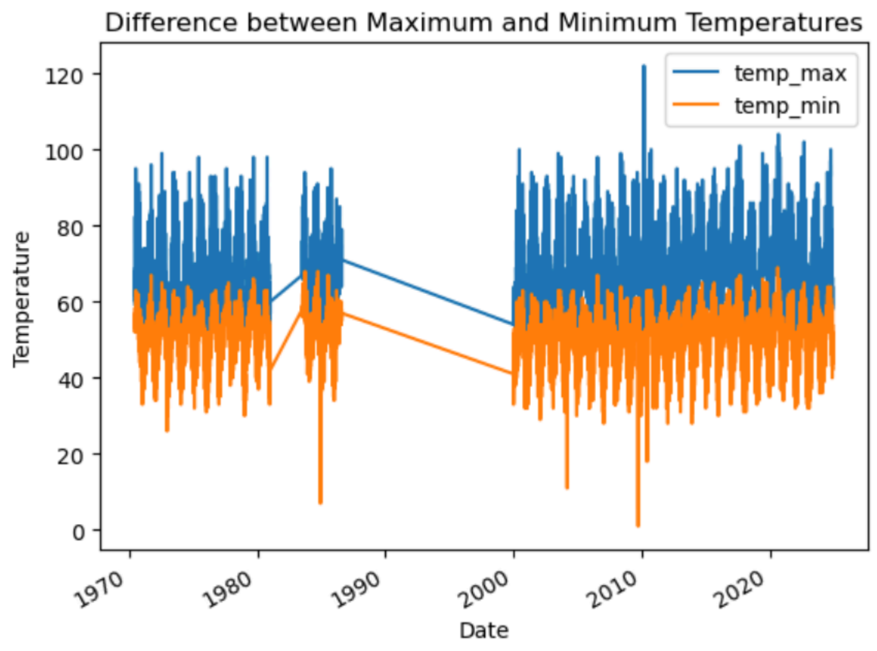
For this, I used the GHCN (Global Historical Climatology Network)-Daily database that provides valuable meteorological information, including precipitation, snowfall, snow depth, maximum temperature, and minimum temperature as core features, along with other information as well *(GHCN-Daily)*. I mainly used these five core features for my prediction. The dataset has data populated at Oakland International Airport from 1970 to 2024 for each day of the year, so it has over 14,000 rows and 10 columns. By analyzing this data, we can gain insights into historical weather patterns, climate trends, and potential future weather events.

Visualization and Analysis:

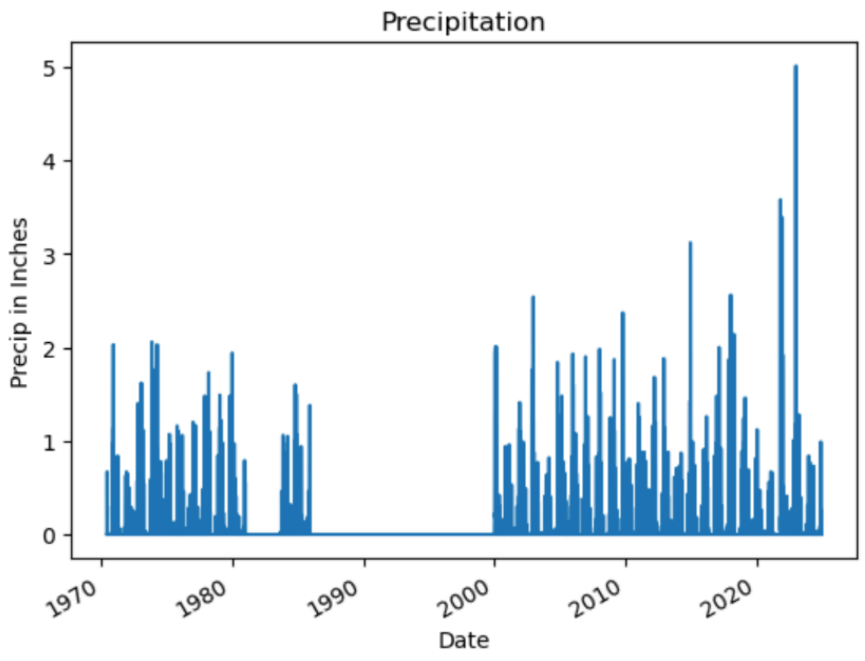
The graph below shows the relationships between maximum and minimum temperatures and plots the average temperature throughout the years. This helps us visualize the average temperature in Oakland Airport from past history.



This next plot shows the difference between the maximum and minimum temperatures throughout the years so we can visualize the difference in the temperatures. The X-axis shows the dates, and the Y-axis shows the temperature in Celsius.



The following graph shows the precipitation through the years. The X-axis shows the dates, and the Y-axis shows the Precipitation in inches.



Analysis:

By looking at these graphs we can see that there are some gaps in them so we're missing data from 1985 to possibly 1999. By using index.year.value counts we can find what points exactly we are missing. This will tell us how many observations exist for each year. So we can see where we're missing data from the years 1981 to 1983 and some data for 1985- 2000 were also missing. This may be because the dataset does not have all the information. It has very few outliers like where the temperature is over 120 which is impossible. But it still has a good amount of historical data that we can use for the prediction.

Data Preprocessing:

The model will not perform well if it has any missing values, so the first thing I did was fix any empty values. I started by finding a column with missing entries that show "NaN". By using the apply method on the weather data and passing it to the pandas in the null function, search all the columns in the weather data to look for any null values and replace them with zeros (*Dataquest*). Then, I deleted all the columns that had many null values, like snow and snow depth, since there is no snow in California, from the core features. Then I also made sure to fill in any missing values with zero as the documentation of the database said any values with nines in a field indicate missing data or data that has not been received so we should make sure to check for that as well.

Method:

The goal is to predict the next day's maximum temperature. For this, I tried to compare Ridge and Linear Regression algorithms and I also tried to KNN to classify the weather type as whether hot or cold based on data collected.

Linear Regression:

Linear Regression is one of the simplest and most widely used algorithms for predicting a continuous target variable based on one or more predictor variables. In this case, since I am trying to predict temperature, which is a continuous variable linear regression seemed like a good choice. It assumes that there is a linear relationship between the predictors and the target variable. However, it does not add any regularization term which can lead to overfitting if the data has multicollinearity or noise. Linear regression is also easy to implement and easy to understand the process and analyze the results we get from it.

Ridge Regression

Ridge Regression is an extension of Linear Regression that aims to address issues related to multicollinearity by adding a penalty to the model. This penalty helps to prevent the model from overfitting. The function that we use from the scikit learn package helps us set a variable lamba which will be the learning rate for the model. It is the regularization parameter that controls the strength of the penalty. The regularization term discourages large coefficients, which helps prevent overfitting by making the model simpler and less sensitive to noise in the training data. So this model might give us better results than Linear regression since it gives us regularization.

K-Nearest Neighbours

K-Nearest Neighbors (KNN) is an algorithm used for classification and regression. It classifies new data points based on the majority class among its K-nearest neighbors. KNN uses distance metrics (typically Euclidean distance) to determine the similarity between data points. The distance function calculates how close or far a point is from others in the feature space. The parameter K determines the number of neighbors to consider. A small K makes the model sensitive to noise, while a large K can smooth out the decision boundary but may miss finer distinctions. Here we will be using it to classify between hot and cold based on the maximum and minimum temperature.

Results

Setup:

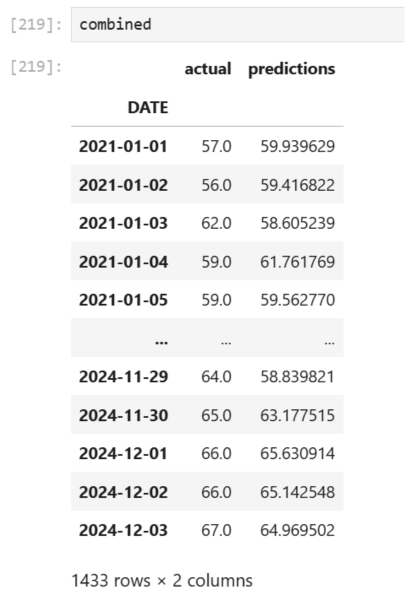
1. We have already seen the data cleaning and preprocessing to prevent any problems.
2. Next, the predictor variables(temperature and precipitation) are decided based on their relation to the target variable.
3. For KNN and Ridge and Linear Regression, we set up two potential types of targets—classification (sunny/rainy) and regression (temperature values). The model will be evaluated based on how accurately it predicts these targets.
4. Then the data is split into training and testing data. The training data is set for the year 2020 and the testing is set for the year after that.

5. I use the Scikit learn functions for all three algorithms.
6. Linear Regression is a basic model with no regularization to predict continuous values based on the input features.
7. Ridge Regression is implemented with the regularization parameter, alpha, that can be adjusted to control the strength of regularization.
8. KNN is used to classify or predict values based on the distance between test data points and the training set to separate them as hot or cold. When the minimum temperature is less that 45 it will be considered cold.
9. Performance is based on the accuracy and Mean squared error between the actual and predicted values. Results are also visualized by graphs.

Test Results of Algorithms and Observations

Results from Ridge Regression:

Here is a picture of a table of the difference between the actual temperature versus the temperature that the model has predicted. There is also a graph that shows the difference between the actual and predicted values for temperature.

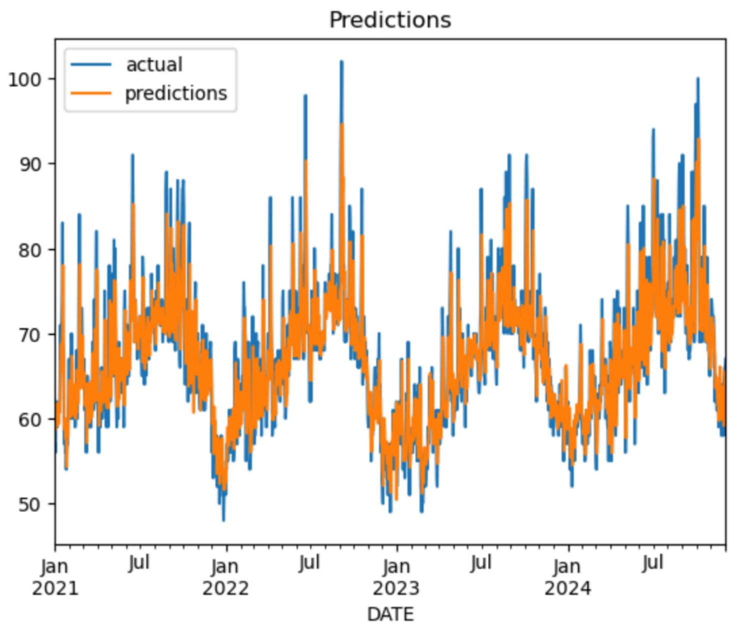


```
[219]: combined
```

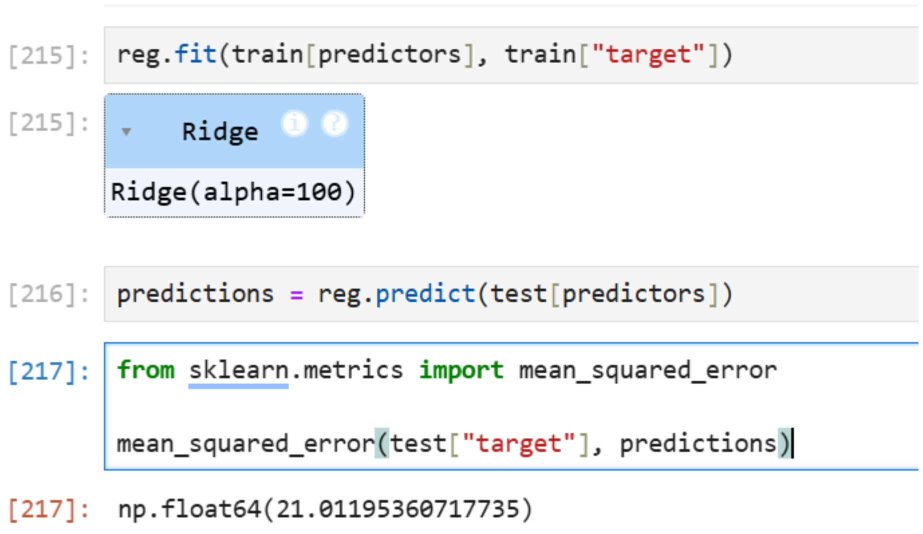
[219]:

DATE	actual	predictions
2021-01-01	57.0	59.939629
2021-01-02	56.0	59.416822
2021-01-03	62.0	58.605239
2021-01-04	59.0	61.761769
2021-01-05	59.0	59.562770
...	...	...
2024-11-29	64.0	58.839821
2024-11-30	65.0	63.177515
2024-12-01	66.0	65.630914
2024-12-02	66.0	65.142548
2024-12-03	67.0	64.969502

1433 rows × 2 columns



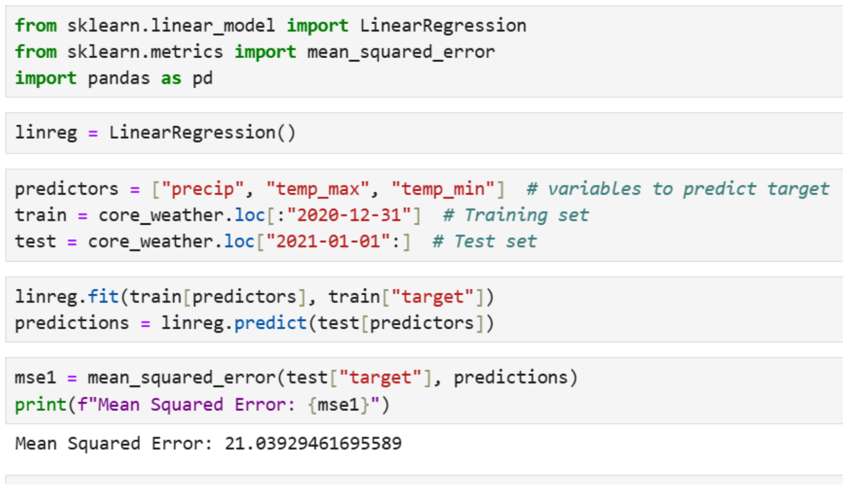
Here we have the learning rate as the Ridge regression model as 100 as the value of “alpha” increases the model shows a lower Mean squared error, which in this case is 20.0119



```
[215]: reg.fit(train[predictors], train["target"])

[215]: Ridge
       Ridge(alpha=100)

[216]: predictions = reg.predict(test[predictors])

[217]: from sklearn.metrics import mean_squared_error

       mean_squared_error(test["target"], predictions)

[217]: np.float64(21.01195360717735)
```

Result from Linear Regression:

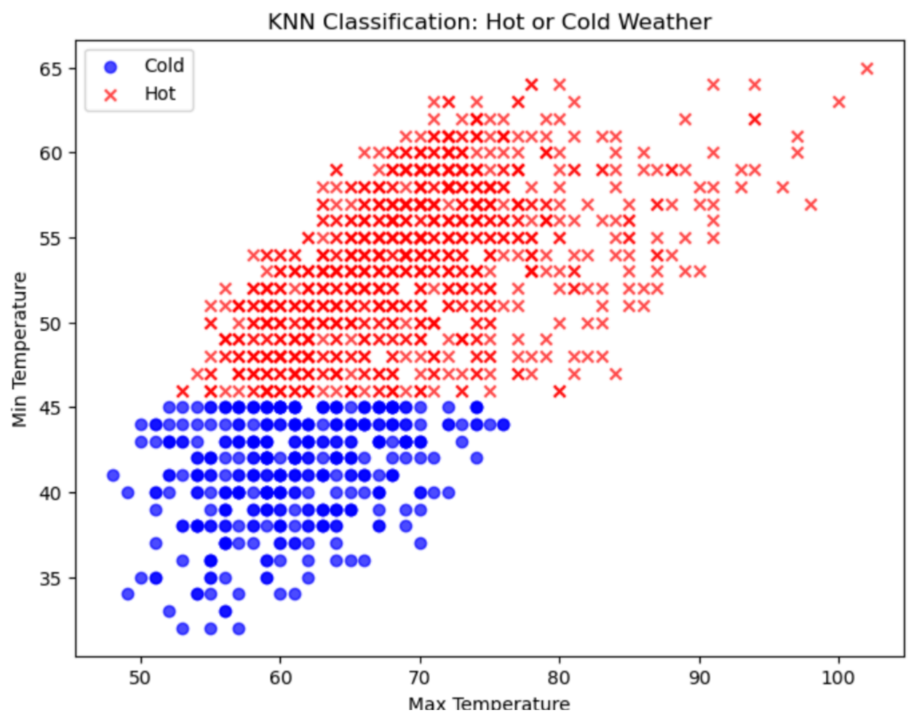
Here we have the Error from the regular Linear Regression model where the error values is slightly higher than the error in the Ridge Regression model with a value of 21.039. They have a difference of about 0.028. This varies based on the values of the regularization parameter used for Ridge Regression as mentioned before.



```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import pandas as pd

linreg = LinearRegression()

predictors = ["precip", "temp_max", "temp_min"]  # variables to predict target
train = core_weather.loc[:"2020-12-31"]  # Training set
test = core_weather.loc["2021-01-01":]  # Test set

linreg.fit(train[predictors], train["target"])
predictions = linreg.predict(test[predictors])

mse1 = mean_squared_error(test["target"], predictions)
print(f"Mean Squared Error: {mse1}")

Mean Squared Error: 21.03929461695589
```

Results from KNN:

The scatter plot will show data points based on the temperature. When the minimum temperature is less than 45 degrees it is considered as a colder day in Oakland. Here is the screenshot that shows that result.



Analysis/Discussion About the Results

The test results reflect the performance of each model in predicting the target variable on the unseen test data.

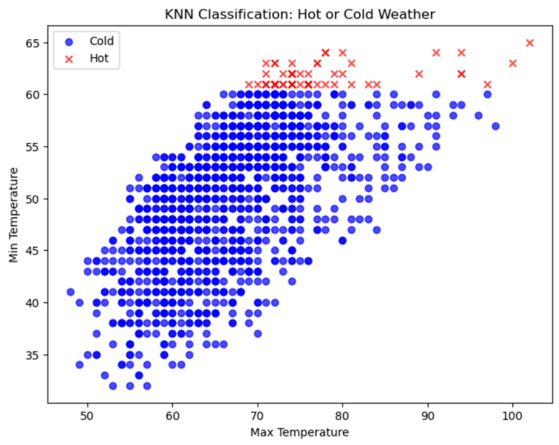
Ridge Regression tends to provide better generalization than standard linear regression due to the regularization, especially when the dataset contains noise or multicollinearity. We see that the error for ridge regression is lesser than that of Linear Regression. The mean squared error of the Ridge regression is 20.011 and that of Linear Regression is 20.039. As the regulartization parameter increases, the error slowly decreases.

For the large part, there is not a big difference between the two algorithms because the data is very consistent and there is not too much noise in the dataset or multicolinearity. When multicollinearity is present, Linear Regression may overfit the training data. The model might have high variance, especially if the dataset includes noisy features or features that are highly correlated.

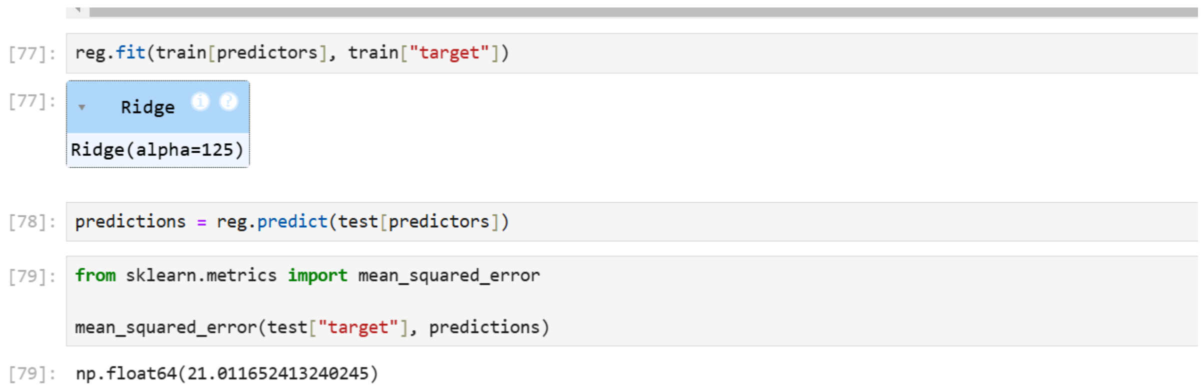
For KNN the model classifies weather based on the feature values (temperature minimum and maximum) that most closely resemble the training data's known weather categories. The more similar the feature values of the test point are to a particular class, the more likely it is to be classified in that category. For weather classification (hot/cold), the test accuracy is quite high, around 0.9.

Experiments:

Let us try to tweak the conditions of the KNN to decide that if the temperature minimum is less than 60 degrees it is cold instead of 45 degrees. We can see the results below there is a much larger number of blue circles than red crosses now.



Now I will change the alpha value of the Ridge Regression to 125 instead of 100 we can see the error value reducing. It has now become 21.01165 instead of 21.01195, so very small changes.


```
[77]: reg.fit(train[predictors], train["target"])

[77]:  Ridge
       Ridge(alpha=125)

[78]: predictions = reg.predict(test[predictors])

[79]: from sklearn.metrics import mean_squared_error

      mean_squared_error(test["target"], predictions)

[79]: np.float64(21.011652413240245)
```

Conclusion

This project has helped me get some experience with Machine Learning outside the classroom or the homework assignments we have done so far. By working on this weather prediction project I have been able to gain a deeper understanding of how to preprocess and analyze data, evaluate model performance, and visualize results effectively. The hands-on experience solidified concepts that I worked on in this project such as KNN, ridge regression, and linear regression models. This also has helped me explore different functions and libraries.

Key Learnings:

- I learnt the importance of data preprocessing for machine learning, and this helped me learn how to handle missing values.
- Implementing and comparing multiple algorithms gave me insight into their strengths and weaknesses, however, since the data is pretty consistent and linear there was not as big of a difference as I was expecting.
- Visualizing data and results helped me understand the results better and identify quickly if there was anything that was going wrong.

Challenges:

I ran into some issues due to the dataset even though I processed it some data was missing from some years, which led me to run into some value errors. I was able to overcome this by deleting any unnecessary columns of data that were causing trouble. I also had some other syntax errors and some implementation issues in some places but I was able to find resources to help me address those issues and learn about why I was going wrong.

This project enhanced some important concepts from Machine learning and highlighted the importance of experimentation, critical thinking, and problem-solving in extracting meaningful insights from data. Overall, it was a great learning experience.

Acknowledgements:

Some information in this document is from online sources. Each source used has in-text citations and is mentioned in the references section as well.

I implemented the Ridge regression model with the help of an online tutorial video from Dataquest, cited.

I also used the help of Chatgpt to help me fix some errors that I received in my code and for help on how to fix it.

Prompt 1: Why am I getting this error?

```
KeyError Traceback (most recent call last) File
~\anaconda3\envs\itcs-3156\lib\site-packages\pandas\core\indexes\base.py:3805, in
Index.get_loc(self, key) 3804 try: -> 3805 return self._engine.get_loc(casted_key) 3806 except
KeyError as err: File index.pyx:167, in pandas._libs.index.IndexEngine.get_loc() File
index.pyx:196, in pandas._libs.index.IndexEngine.get_loc() File
pandas\\_libs\\hashtable_class_helper.pxi:7081, in
pandas._libs.hashtable.PyObjectHashTable.get_item() File
pandas\\_libs\\hashtable_class_helper.pxi:7089, in
pandas._libs.hashtable.PyObjectHashTable.get_item() KeyError: 'weather_category'
```

Result:
The error KeyError: 'weather_category' occurs because the column weather_category does not exist in your train DataFrame. This might happen for several reasons:

1. Column Name Mismatch: The column weather_category might have a different name (e.g., WeatherCategory, weather category, or something else). Column names in pandas are case-sensitive and need to match exactly.
2. Missing Data: The column might not have been created or included in the core_weather DataFrame. Check whether the column exists in core_weather.
3. Data Filtering Issue: When performing the train-test split using .loc[], ensure the weather_category column was not dropped or excluded inadvertently.

Prompt 2: Why am I getting Value Error value too large for float64?

Output:

The error "Value too large for float64" typically occurs when a value in your dataset exceeds the maximum allowable value for the float64 data type (approximately 1.8×103081.8 \times 10^{308}1.8×10308) or contains an unexpected datatype (like strings or extreme outliers). Here's how to troubleshoot and resolve this issue

Steps to Identify and Fix the Issue

1. Inspect the Dataset for Outliers or Invalid Values:
    - Check for extremely large values or unexpected entries in your dataset.
2. If you see suspiciously high numbers, you might have corrupted data.
3. Look for Non-Numeric Data:
    - If a non-numeric value is inadvertently included, it could lead to this error. Ensure all columns in predictors are numeric (float64 or int64).
4. Handle Infinite or Missing Values:
    - Check for inf, -inf, or NaN values in your dataset
    - If found, you can replace them with suitable values or remove the rows:
    -
    -
5. Validate Data Before Processing:
    - Ensure predictors are within a logical range for weather data:

References:

Council, National Research. “When Weather Matters.” *National Academies Press eBooks*, 2010, https://doi.org/10.17226/12888.

Dataquest. “Predict the Weather With Machine Learning.” *YouTube*, 10 Feb. 2022, www.youtube.com/watch?v=km95-NMT6lU.

Global Historical Climatology Network - Daily (GHCN-Daily), www.ncei.noaa.gov/data/daily-summaries/doc/GHCND_documentation.pdf.

Ritchie, Hannah, and Max Roser. “Weather Forecasts Have Become Much More Accurate; We Now Need to Make Them Available to Everyone.” *Our World in Data*, 15 Mar. 2024, ourworldindata.org/weather-forecasts.