SJSU SAN JOSÉ STATE UNIVERSITY

ML Weather Prediction

Jiaying Li
Advisor: Maryam Khazaei Pool

Introduction

Weather forecast is important in many ways, accurate forecasting can give warnings for storms, heat waves, and potential disasters. These factors can be used by people in all kinds of different fields and also in daily life.

Our goal is to be able to make a reasonable prediction on the next day’s weather based on publicly accessible weather data.

We wanted to practice ways in data processing and cleaning, thus learning more about machine learning and its implementation.

Problem Statement & Contribution

Using Machine Learning for weather prediction.

Specifically, to implement a Linear Regression Model trained with weather data to predict the next day's temperature with an acceptable accuracy.

Reference: Holmstrom, Mark, Dylan Liu, and Christopher Vo. "Machine learning applied to weather forecasting." Meteorol. Appl 10.1 (2016): 1-5.

Timeline

01

Data Collection and Processing:

- Collect/compile weather data
- Pre-process and organize data
- Clean data

02

Statistical Analysis:

- Discover correlation
- Run variable selection
- Select relevant statistical graphs

03

Model Implementation:

- Create relevant graphs
- Implement prediction model(s) with selected variables

04

Accuracy Check:

- Run predictions to testing data
- Check R-square and draw conclusion
- Revise model if needed

Basic Process

Data Processing

- Google Sheets
- Python

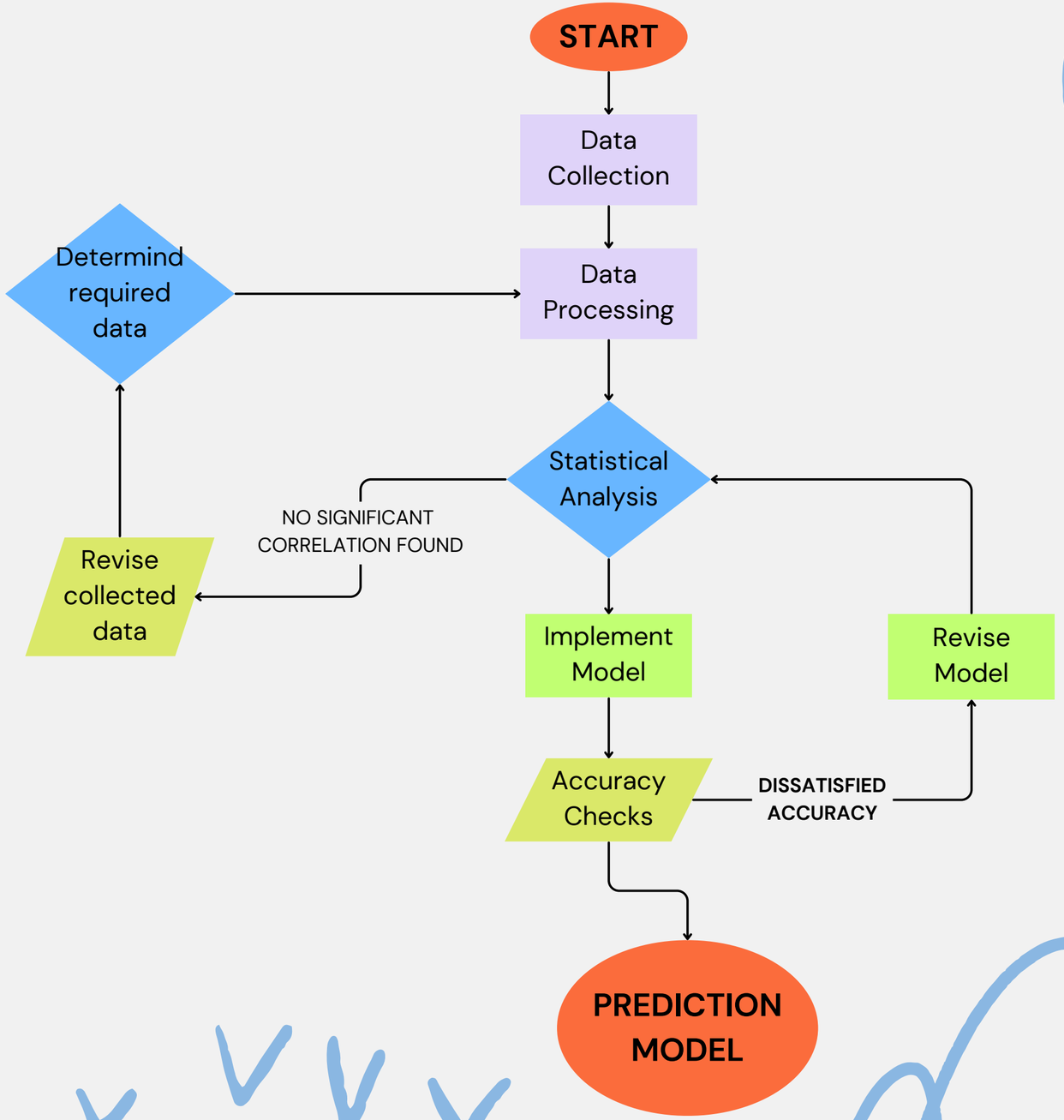
Statistcal Analysis

- JMP statistical software
- Python

Model Implementation

- Python Pandas package
- Python sklearn package

Weather Prediction Model



Data

Data Collection, Processing, and Cleaning

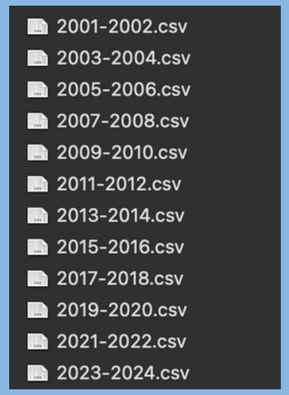
Data Source

A source for historical and forecast weather data.

Python Libraries

```
#import libraries needed
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from scipy import stats
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
```

Data Collecting



Compiled more than two decades worth of historic San Jose weather data.

Basic Cleaning

```
#drop duplicated rows based on date
data_cleaning.drop_duplicates(inplace=True)
```

```
# drop NA rows
data_cleaning.dropna(inplace=True)
```

Droppped irrelevant columns in Google Sheets.

After completing the baisc cleanings, we are left with only 12 years worth of weather data.

Basic Organization

Checked uniqueness of dates.

```
=UNIQUE(A2:A, true, true)
```

fulldate	year	month	season
2010-01-01	2010	01	winter
2010-01-02	2010	01	winter
2010-01-03	2010	01	winter

Added Year, Month, and Season columns.

Got number of hours the sun is up.

```
=TEXT((right(AI2, 8))-(right(AH2, 7)), "h:mm:ss")
```

	AH	AI	AJ	AK
	sunrise	sunset	sunduration	sunduration
	2010-01-01 7:21:55	2010-01-01 17:00:41	9:38:46	9.3846
	2010-01-02 7:22:03	2010-01-02 17:01:30	9:39:27	9.3927
	2010-01-03 7:22:09	2010-01-03 17:02:19	9:40:10	9.4010
	2010-01-04 7:22:13	2010-01-04 17:03:11	9:40:58	9.4058
	2010-01-05 7:22:15	2010-01-05 17:04:03	9:41:48	9.4148

Remove Outliers

```
print("Range for temperature data: ")
print("temp: ", data_cleaning['temp'].min(), ", ", data_cleaning['temp'].max())
clean_data = data_cleaning[(np.abs(stats.zscore(data_cleaning['temp']) < 3))]
print("\nRange for temperature data after dropping: ")
print("temp: ", clean_data['temp'].min(), ", ", clean_data['temp'].max())
```

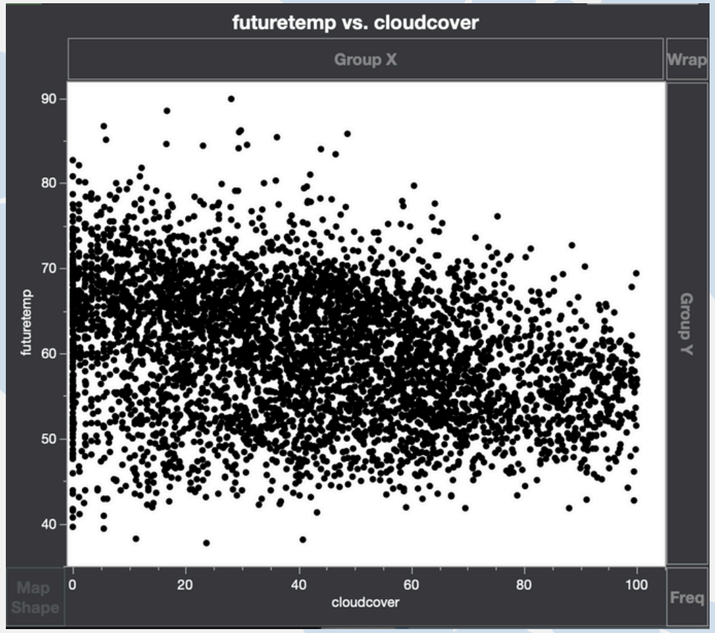
```
Range for temperature data:
temp:  42.7 ,  84.5

Range for temperature data after dropping:
temp:  42.7 ,  70.3
```

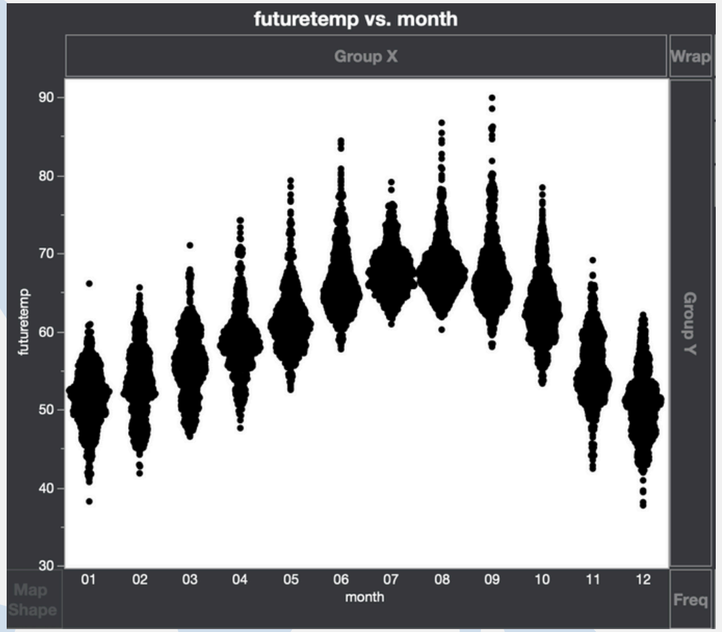
Want to drop any rows with values with a z-score higher than 3.

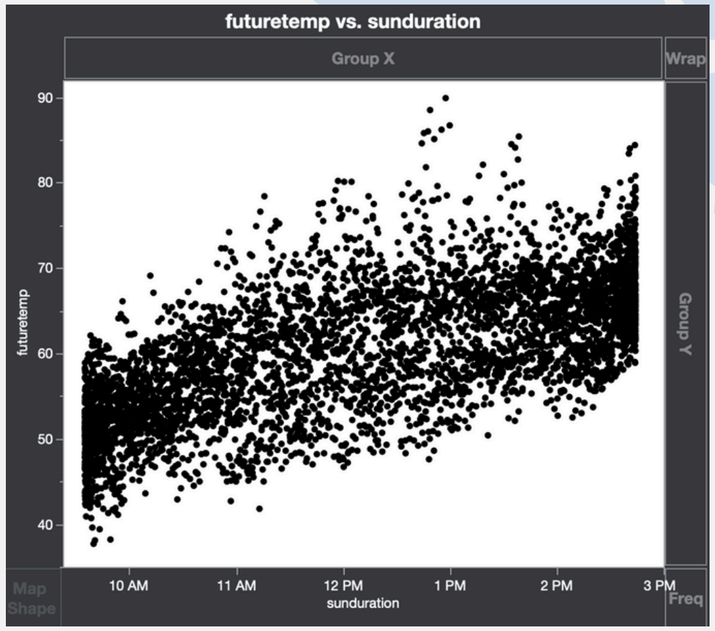
```
Number of duplicated rows dropped: 2
```

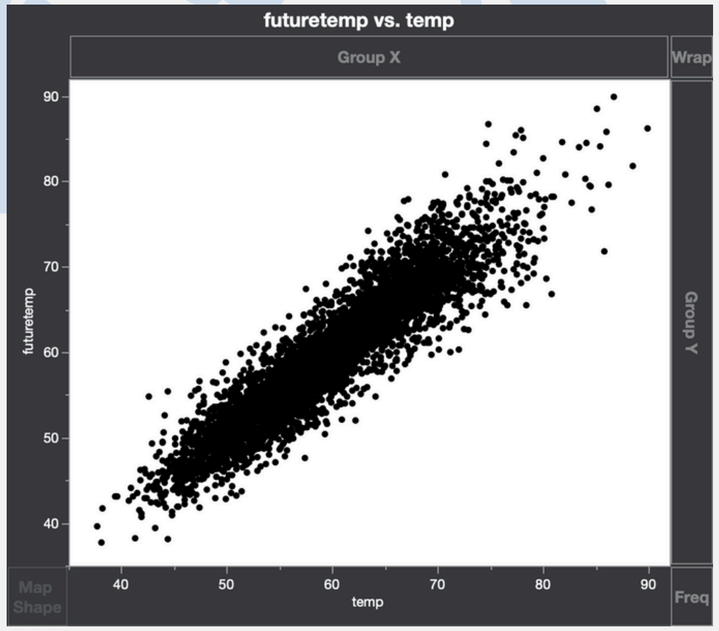
Correlations

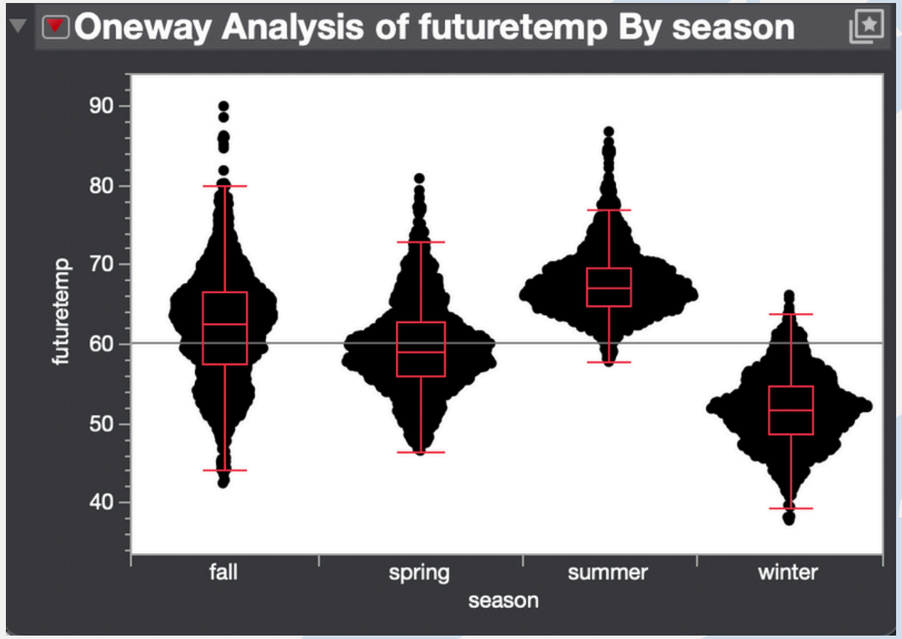
Between next-day's temperature and variables











Variable Selection

Fit Group

Response futuretemp

Effect Summary

Source	Logworth	PValue
temp	811.010	0.00000
1agotemp	78.788	0.00000
sunduration	42.620	0.00000
2agotemp	37.623	0.00000

Remove Add Edit Exclude FDR

Summary of Fit

RSquare	0.87672
RSquare Adj	0.876616
Root Mean Square Error	2.771896
Mean of Response	60.17612
Observations (or Sum Wgts)	4748

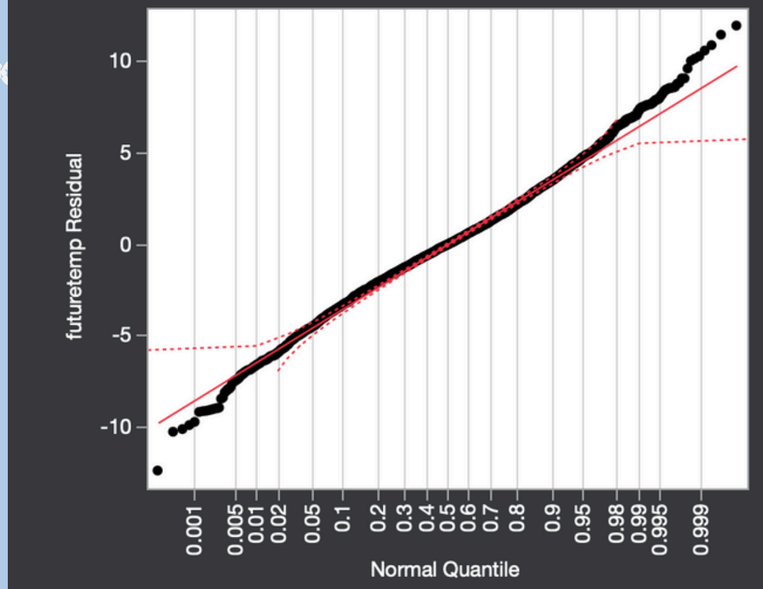
Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	4	259164.90	64791.2	8432.618
Error	4743	36442.39	7.7	Prob > F
C. Total	4747	295607.29		<.0001*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	2.937123	0.330435	8.89	<.0001*
temp	1.07225	0.014249	75.25	<.0001*
1agotemp	-0.391243	0.020332	-19.24	<.0001*
2agotemp	0.1843491	0.01411	13.06	<.0001*
sunduration	0.431261	0.030926	13.94	<.0001*

Residual Normal Quantile Plot



Stepwise Fit for futuretemp

Stepwise Regression Control

Stopping Rule: P-value Threshold

Prob to Enter 0.1

Prob to Leave 0.1

Direction: Mixed

Enter All Make Model

Remove All Run Model

Go Stop Step

Training Rows 4748

SSE	DFE	RMSE	RSquare	RSquare Adj	Cp	p	AICc	BIC
36442.393	4743	2.7718957	0.8767	0.8766	5	5	23162.73	23201.5

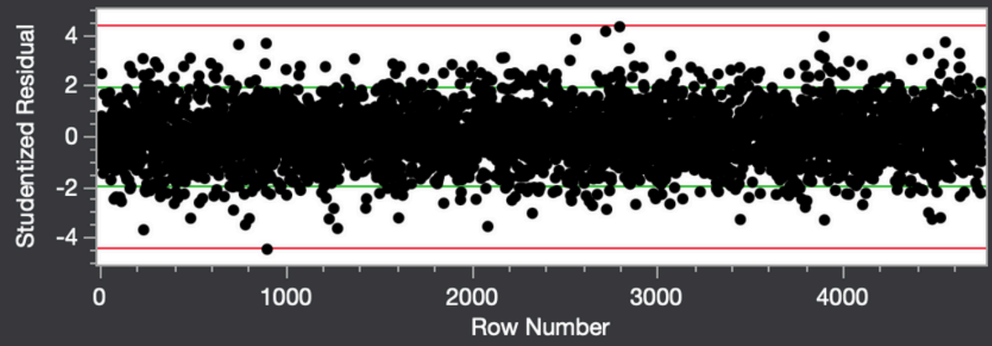
Current Estimates

Lock	Entered	Parameter	Estimate	nDF	SS	"F Ratio"	"Prob>F"
☑	☑	Intercept	2.93712303	1	0	0.000	1
☐	☑	temp	1.07224997	1	43509.35	5662.769	0
☐	☑	1agotemp	-0.3912434	1	2845.004	370.279	1.6e-79
☐	☑	2agotemp	0.18434915	1	1311.481	170.690	2.4e-38
☐	☑	sunduration	0.43126095	1	1494.126	194.461	2.4e-43

Step History

Step	Parameter	Action	"Sig Prob"	Seq SS	RSquare	Cp	p	AICc	BIC
1	temp	Entered	0.0000	254653.5	0.8615	586.16	2	23710.9	23730.3
2	sunduration	Entered	0.0000	1648.75	0.8670	373.58	3	23517.8	23543.6
3	1agotemp	Entered	0.0000	1551.187	0.8723	173.69	4	23328.6	23360.9
4	2agotemp	Entered	0.0000	1311.481	0.8767	5	5	23162.7	23201.5

Studentized Residuals



Externally studentized residuals with 95% simultaneous limits (Bonferroni) in red, individual limits in green.

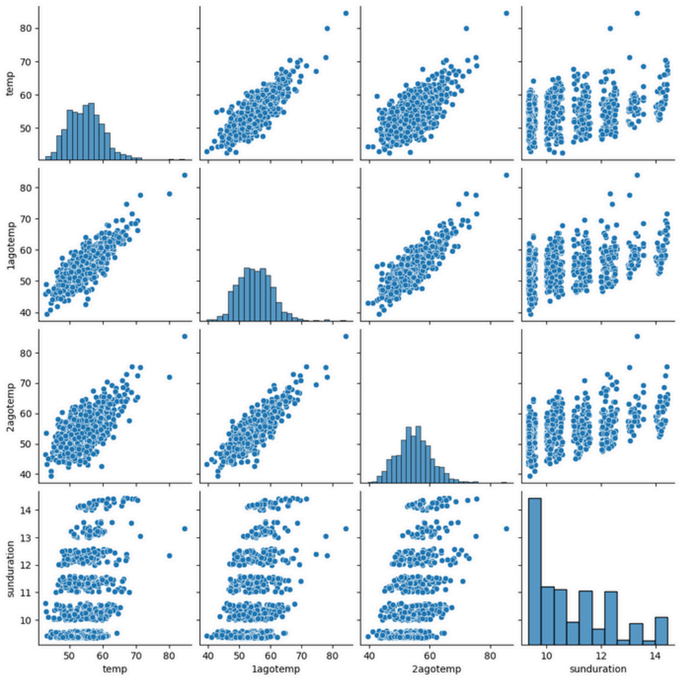
Model Implementation

4 variable Linear Regression Model

```python
# create regression model
lr = LinearRegression()
y = clean_data['futuretemp']
X = clean_data.loc[:, ['temp', '1agotemp', '2agotemp', 'sunduration']]
model = lr.fit(X, y)
```

```python
# get analysis of model
print("Model intercept: ", model.intercept_)
print("Model slope for X1, X2, X3, and X4 respectively: ", model.coef_)
```

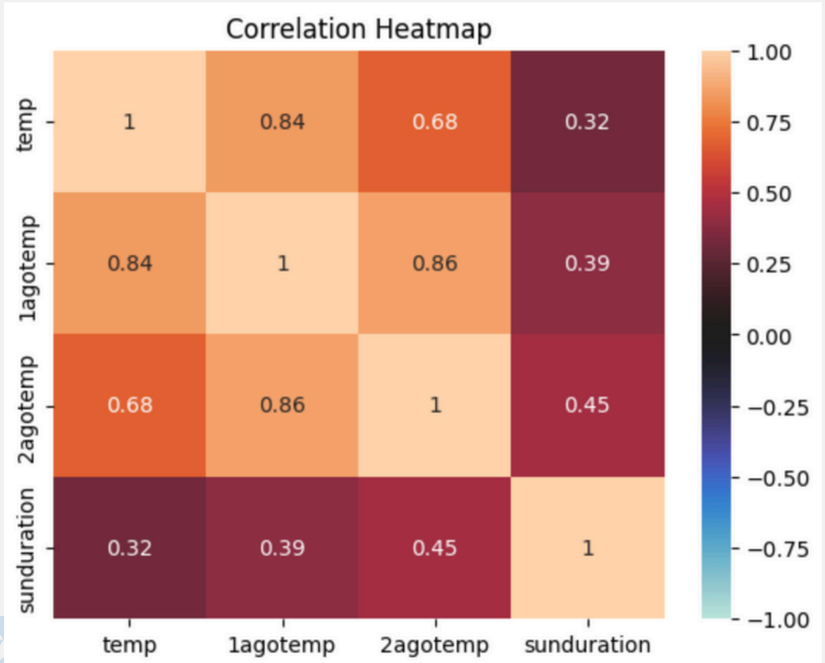
```
Model intercept:  2.9270349822337636
Model slope for X1, X2, X3, and X4 respectively:  [ 1.02423723 -0.35365977  0.15700545  0.53375115]
```




```python
# pair plot of the 4 variables
sns.pairplot(X)
plt.show()
```

```python
# correlation between variables
X.corr()
```

	temp	1agotemp	2agotemp	sunduration
temp	1.000000	0.840278	0.675441	0.318914
1agotemp	0.840278	1.000000	0.863774	0.389378
2agotemp	0.675441	0.863774	1.000000	0.445543
sunduration	0.318914	0.389378	0.445543	1.000000




```python
# heat map
sns.heatmap(X.corr(), annot=True, vmin=-1, vmax=1, center=0)
plt.title("Correlation Heatmap")
plt.show()
```

Formula: y = *B0* + *B1*X1 + *B2*X2 + *B3*X3

- *B0*: intercept of 2.9270349822337636
- *B1*: temperature slop of 1.02423723
- *B2*: yesterday's temperature slope of -0.35365977
- *B3*: slope for temperature from two days ago is 0.15700545
- *B4*: sun duration slope of 0.53375115

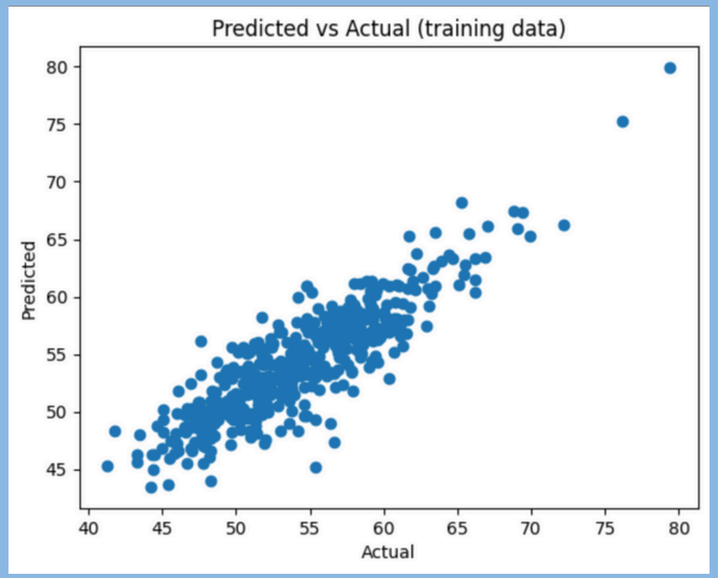
```
# split dataset to training and testing datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

Model Training

```
model = lr.fit(X_train, y_train)
```

```
# get predicted y
y_train_pred = model.predict(X_train)
```

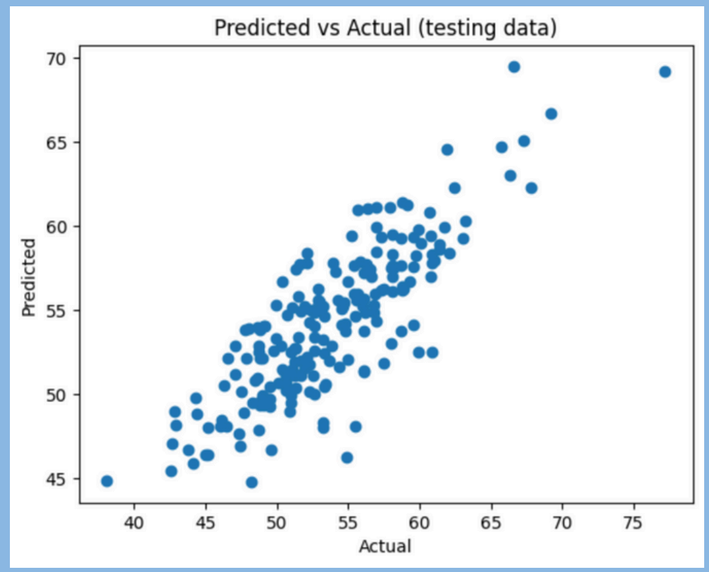
```
# plot correlation between predicted and actual
plt.plot(y_train, y_train_pred, 'o')
plt.title("Predicted vs Actual (training data)")
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.show()
```



Model Testing

```
# predict future temperature based on testing data
y_test_pred = model.predict(X_test)
```

```
# plot correlation between predicted and actual
plt.plot(y_test, y_test_pred, 'o')
plt.title("Predicted vs Actual (testing data)")
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.show()
```



Accuracy Check

Model accuracy:

68%

With an average over/under estimate of 2.453 degrees Fahrenheit

```
# get R-square
print("Prediction accuracy (R-square) on training data", model.score(X_train, y_train))
```

```
Prediction accuracy (R-square) on training data 0.766655588024369
```

```
mae_train = mean_absolute_error(y_train, y_train_pred)
print("Training data MAE: ", mae_train)
```

```
Training data MAE:  2.044144819273232
```

```
# get R-square of prediction
print("R-square on testing data", model.score(X_test, y_test))
```

```
R-square on testing data 0.6804222535319139
```

```
mae_test = mean_absolute_error(y_test, y_test_pred)
print("Testing data MAE: ", mae_test)
```

```
Testing data MAE:  2.452781988229112
```

Limitation and Future Works

Due to the nature of a linear regression model, the model may not be able to predict unusual weather phenomenas. Resulting in modelt conservative predictions in extreme cases.

One possible improvement for this model could be to include cloud and climatology data. With data on how the clouds have moved and been, we will be able to make a reasonable prediction on how cloud data will be for the next day. Hence, using this as an additional variable for our regression model.

Conclusion

The predictive model works consistently. The data of the variables used in the model has been cleaned and processed. So one can used this model for easy next day temperature prediction.

This model is easy to use since the variables needed for running the prediction is very accessible. These variables include today's temperature, today's sun duration, yesterday's temperature, and temperature from two days ago.
Another idea to increase the accuracy of the model could be to consider ANOVA and variable interactions, but this should be done with model complexity and increase of accuracy in mind.

Thank you very much!