

Project Report:

By: Nishant, John and Liam

Pre-Processing

As part of project requirements our team has been hired to analyze weather data of Australia. After this project we can easily describe about the weather data variation across all states of Australia.

The following table illustrates the observations and actions taken by the group to ensure a clean data set.

Pre-process Step	Data Need	Observation	Action
1.	100+ rows of data needed	Data well observed and relevant to project task	csv file added in resources
2.	Sql database	Weather data is stored with the weatherob database	Part of data cleaning job
3.	Weather.js, weathermap.js, weather.min.js, leaflet-heat.js, app.js	Main JavaScripts	Added this scripts in static folder
4.	Jquery.js	jQuery library	As per project requirements
5.	Config.py	insert Username = <'username'> and Password = <'password'>	Add with .gitignore
6.	Project Proposal	Add all requirements, Data sources and brief descriptions about project	Project proposal is added to GitHub
7.	Dependencies	import pandas as pd, from sqlalchemy import create_engine, import config, import requests,	

Pre-process Step	Data Need	Observation	Action
		import numpy, from splinter import browser, from bs4 import beautiful soup as bs, import time, import datetime, import flask, import web driver manager. Chrome, import random, import sqlite3	
8.	App.py	Python file	Indicates all route of API

Introduction

The purpose of this project is to identify the correct weather variation across different states of weather stations of Australia. Here we combine API routes with java script and based of the data we got we generate heat map and mean accuracy of all weather stations data. Here we used SQL with weatherobs database.

We used python flask api with different routes and connect that route with java script and create webpage with maps and charts.

Here we created dropdown menu for all states of Australia. We created heatmap and when we zoom in map we can easily distinguish the weather variation of all weather stations.

Flask API routes:1. /api/v1.0/stationdata

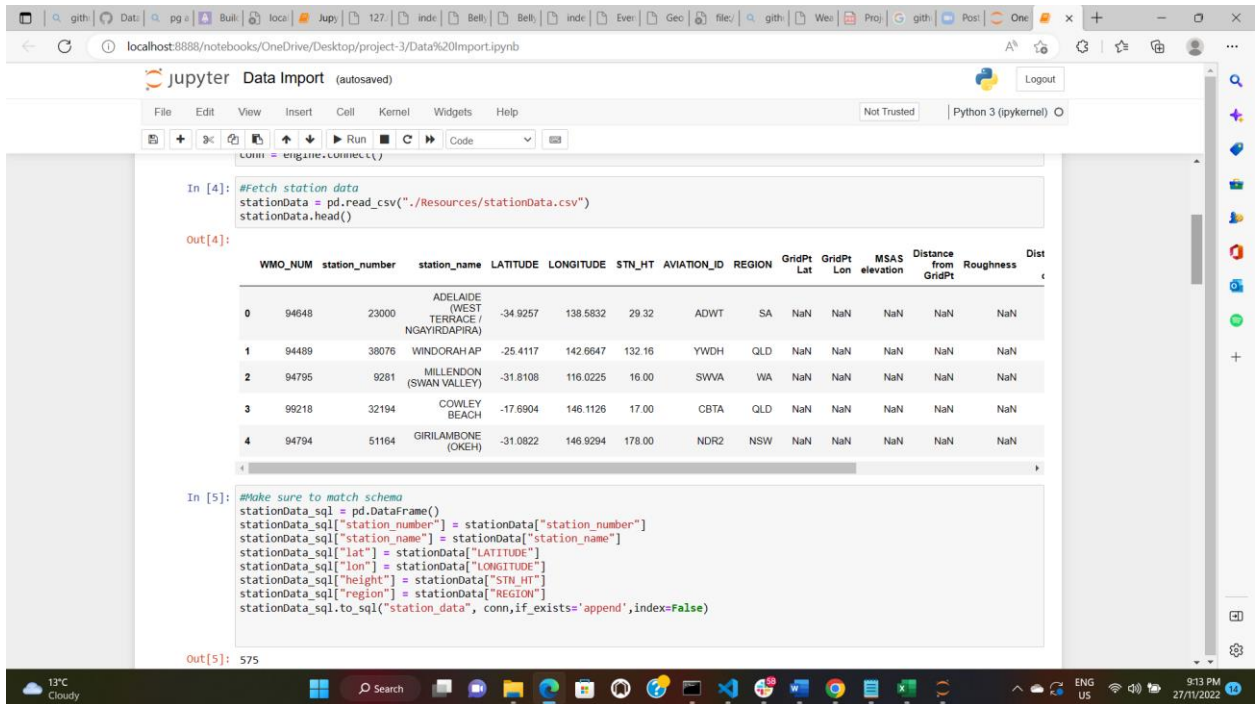
2. /api/v1.0/fcst

3. /api/v1.0/obs

4. /api/v1.0/var

Data cleaning

We start our project with data cleaning process. We cleaned the datasets and remove all duplicates from datasets. We used pandas read function to fetch data from csv. Then we stored all data with new column name. Here we attached screenshot of read data from csv file. Then we stored data in SQL data base with weatherobs database.



The screenshot shows a Jupyter Notebook interface with the following code and output:

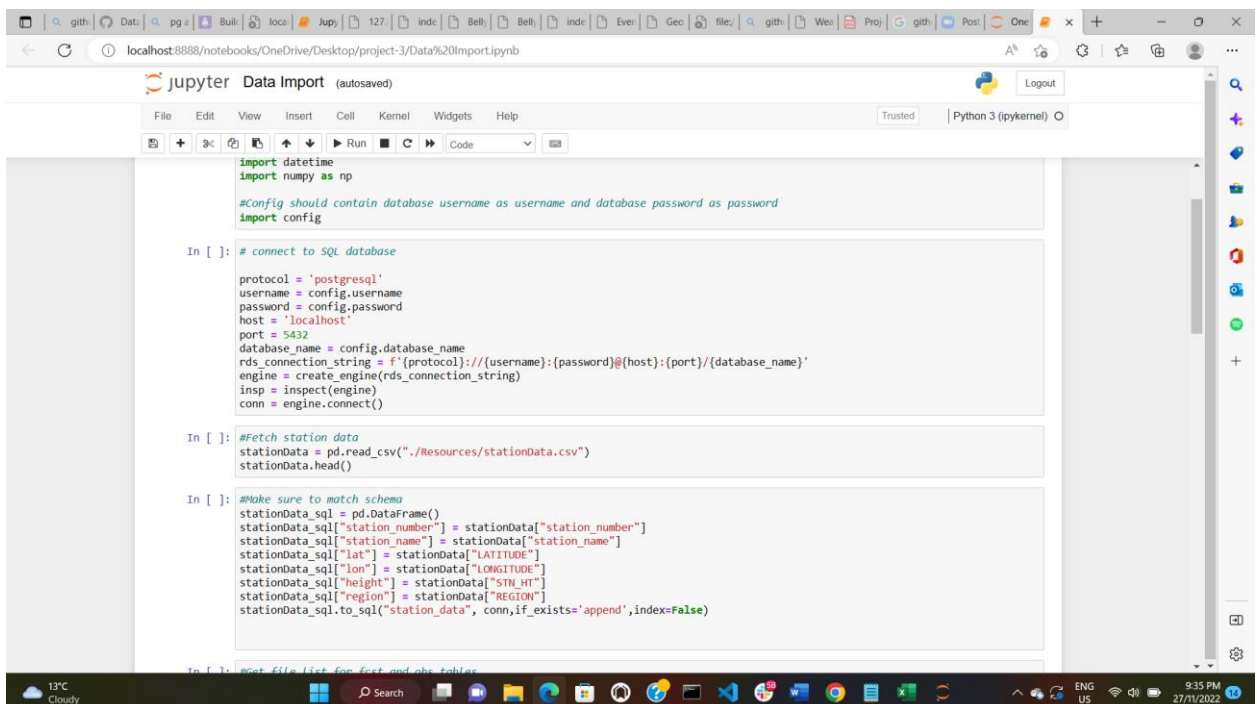
```
In [4]: #Fetch station data
stationData = pd.read_csv("../Resources/stationData.csv")
stationData.head()

Out[4]:
```

	WMO_NUM	station_number	station_name	LATITUDE	LONGITUDE	STN_HT	AVIATION_ID	REGION	GridPt Lat	GridPt Lon	MSAS elevation	Distance from GridPt	Roughness	Dist
0	94648	23000	ADELAIDE (WEST TERRACE / NGAYIRDAPIRA)	-34.9257	138.6832	29.32	ADWT	SA	NaN	NaN	NaN	NaN	NaN	
1	94489	38076	WINDORAH AP	-25.4117	142.6647	132.16	YWDH	QLD	NaN	NaN	NaN	NaN	NaN	
2	94795	9281	MILLENDON (SWAN VALLEY)	-31.8108	116.0225	16.00	SWVA	WA	NaN	NaN	NaN	NaN	NaN	
3	99218	32194	COWLEY BEACH	-17.6904	146.1126	17.00	CBTA	QLD	NaN	NaN	NaN	NaN	NaN	
4	94794	51164	GIRILAMBONE (OKEH)	-31.0822	146.9294	178.00	NDR2	NSW	NaN	NaN	NaN	NaN	NaN	

```
In [5]: #Make sure to match schema
stationData_sql = pd.DataFrame()
stationData_sql["station_number"] = stationData["station_number"]
stationData_sql["station_name"] = stationData["station_name"]
stationData_sql["lat"] = stationData["LATITUDE"]
stationData_sql["lon"] = stationData["LONGITUDE"]
stationData_sql["height"] = stationData["STN_HT"]
stationData_sql["region"] = stationData["REGION"]
stationData_sql.to_sql("station_data", conn, if_exists='append', index=False)

Out[5]: 575
```



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
import datetime
import numpy as np

#Config should contain database username as username and database password as password
import config

In [ ]: # connect to SQL database

protocol = 'postgresql'
username = config.username
password = config.password
host = 'localhost'
port = 5432
database_name = config.database_name
rds_connection_string = f'{protocol}://{username}:{password}@{host}:{port}/{database_name}'
engine = create_engine(rds_connection_string)
insp = inspect(engine)
conn = engine.connect()

In [ ]: #Fetch station data
stationData = pd.read_csv("../Resources/stationData.csv")
stationData.head()

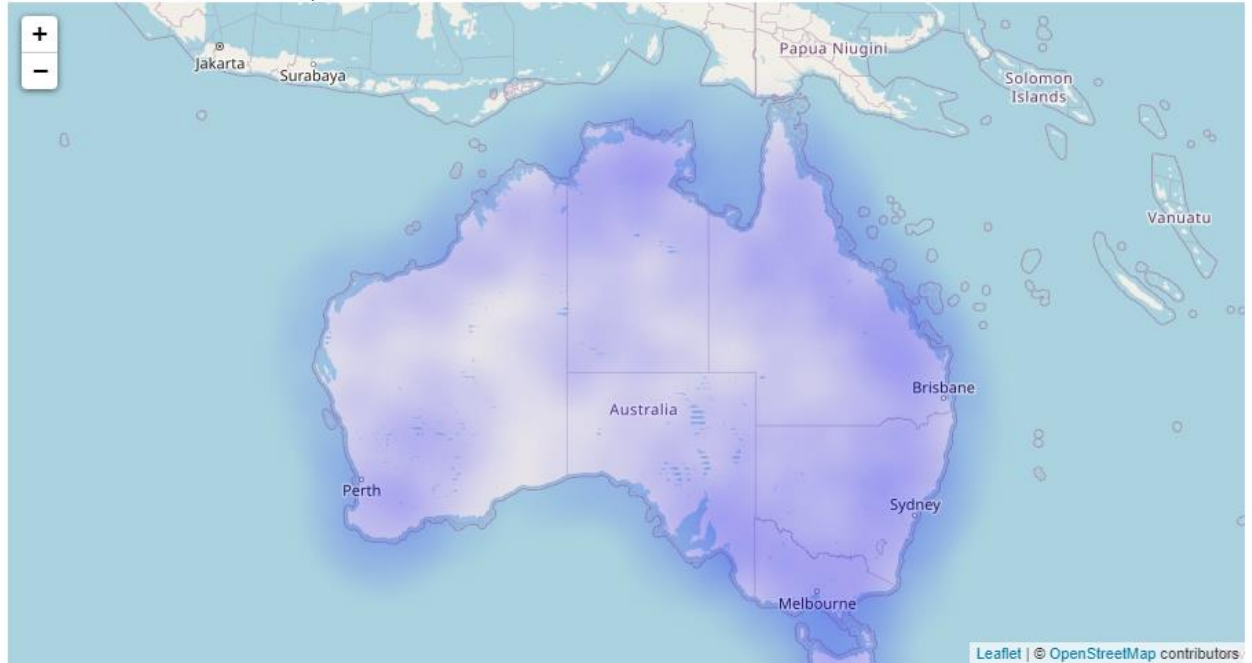
In [ ]: #Make sure to match schema
stationData_sql = pd.DataFrame()
stationData_sql["station_number"] = stationData["station_number"]
stationData_sql["station_name"] = stationData["station_name"]
stationData_sql["lat"] = stationData["LATITUDE"]
stationData_sql["lon"] = stationData["LONGITUDE"]
stationData_sql["height"] = stationData["STN_HT"]
stationData_sql["region"] = stationData["REGION"]
stationData_sql.to_sql("station_data", conn, if_exists='append', index=False)
```

Data processing

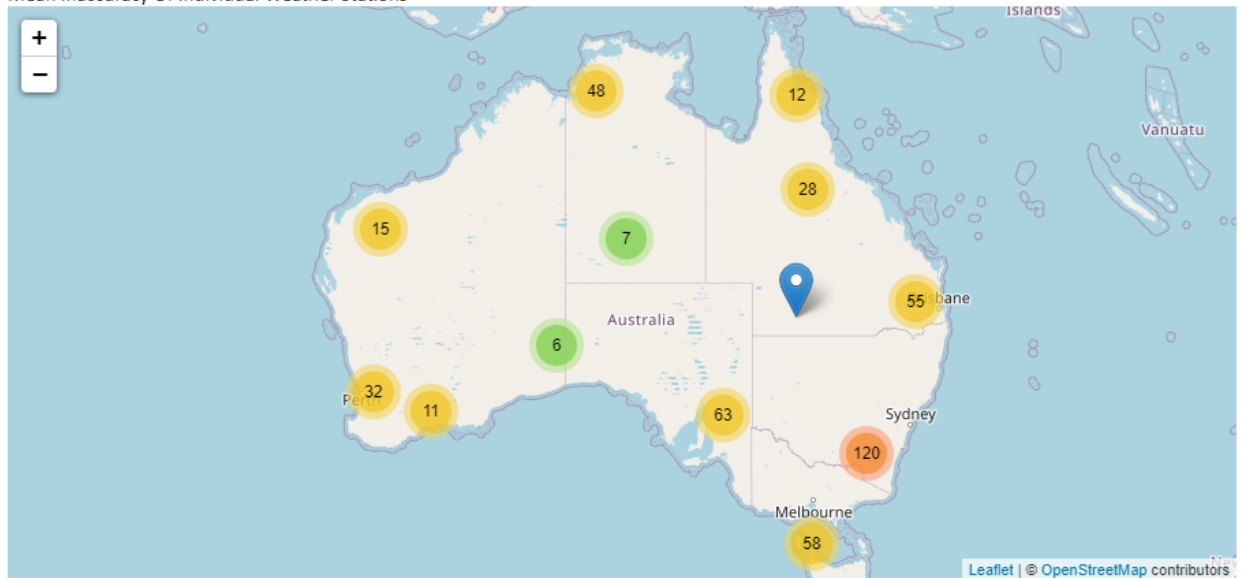
- Connect to the local database. Here create a config.py file and keep your username and password in it and save the config.py file in .gitignore file to keep your username and password confidential. If it's not confidential, you can put it straight away.
- Load csv converted DataFrames into database (weatherobs)
- Confirm data has been added by querying the tables in both pandas and postgresQL
- Here we created different api routes which we explained in introduction and using that route we fetch station data and then we performed java scripts and we generate heat map with weather accuracy variation of different weather station data.
- We used jquery java library to perform this task.
- Here we used a visualisation of the maximum and minimum temperature across the week, with some uncertainty based on the historical uncertainty of stations in your region.
- There will be a projection of the hourly temperature on that day, based on the historical hourly temp pattern for that day in your area.
- There will also be a map of the uncertainty in weather station temperature across the globe.
- The data processing and web scraping for this dashboard will be powered by Flask, Leaflet will be used for the mapping and Plotly for the charts
- We used JS library for weather queries and we used Flask for making database queries and then we used webpage HTML for final visualisations.

Outputs Screenshots

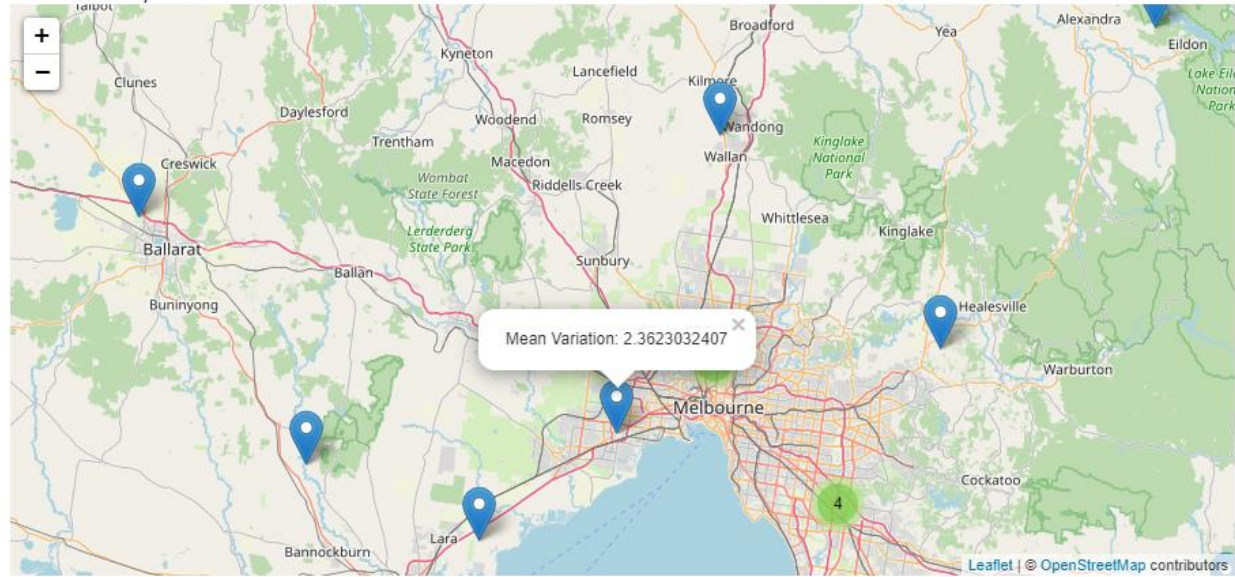
Weather Observation Station Density Across Australia



Mean Inaccuracy Of Individual Weather Stations



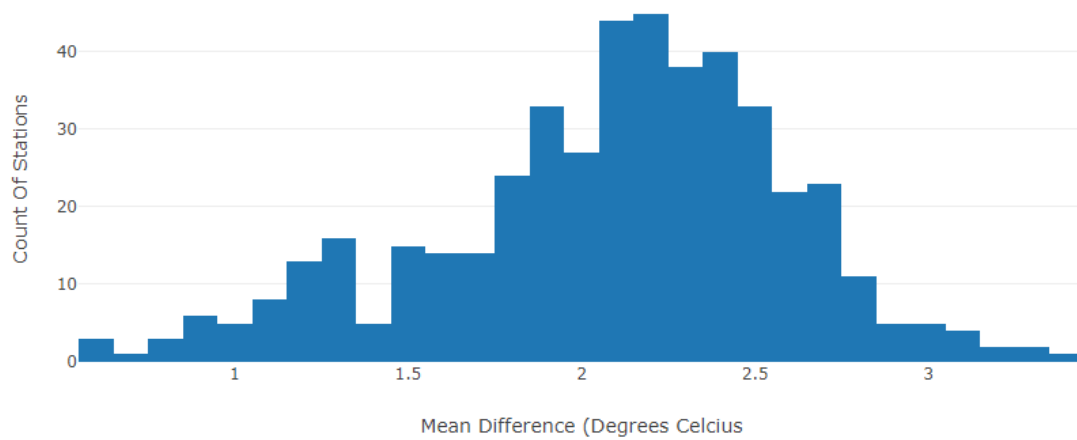
Mean Inaccuracy Of Individual Weather Stations



State:

All

Mean Difference From Observed Temp

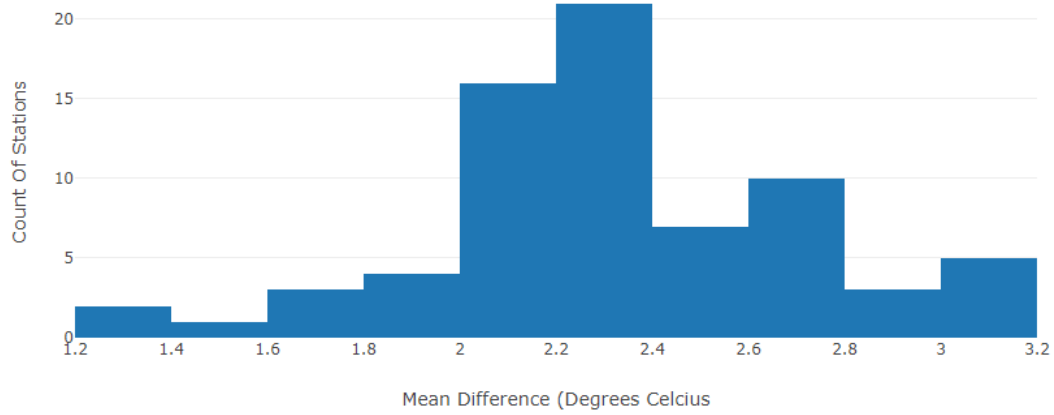


State:

VIC



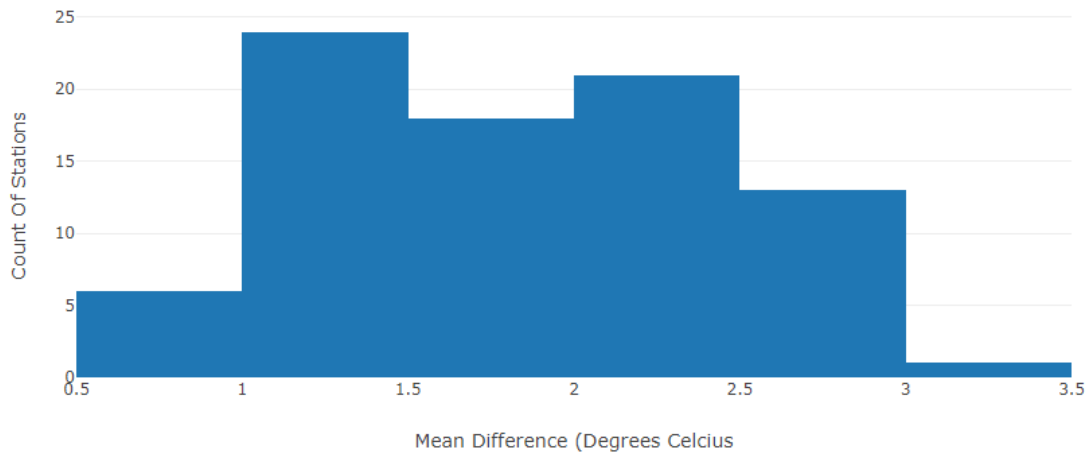
Mean Difference From Observed Temp



State:

QLD

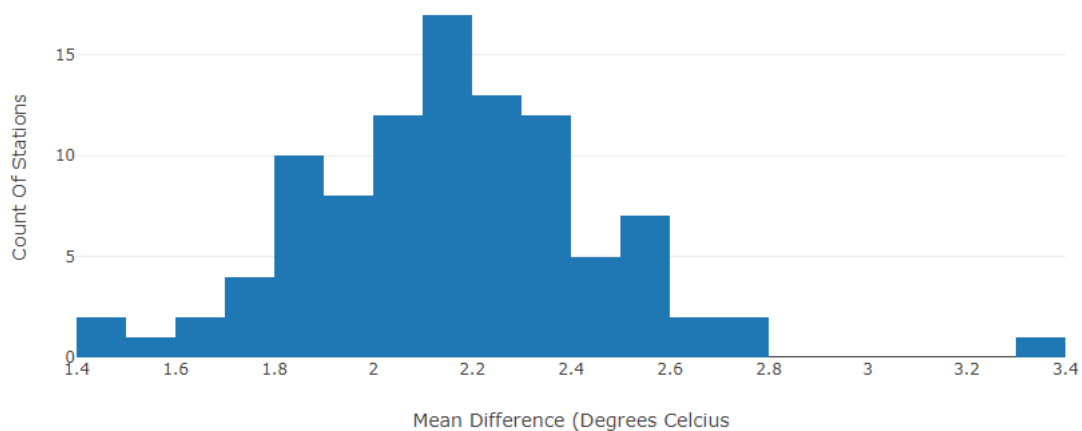
Mean Difference From Observed Temp



State:

NSW

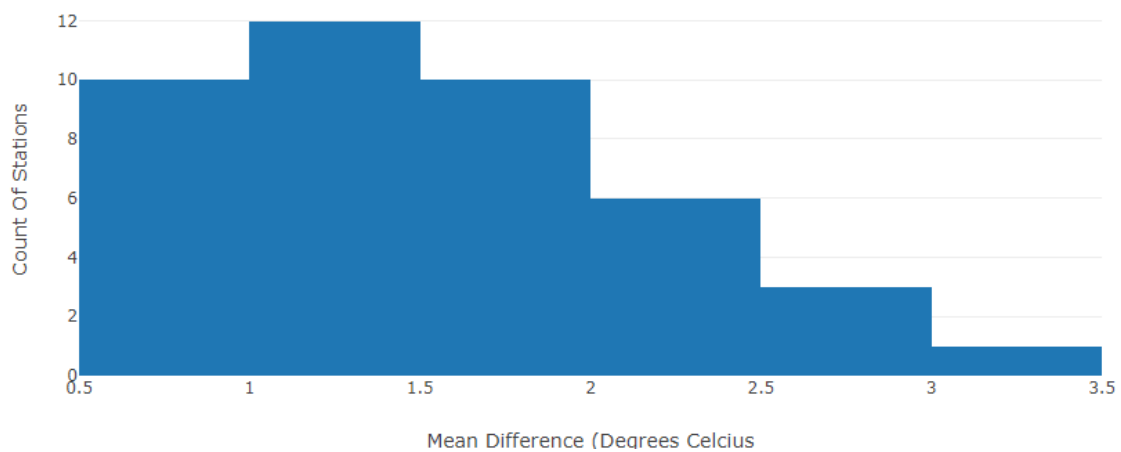
Mean Difference From Observed Temp



State:

NT

Mean Difference From Observed Temp



Summary

Here we made some findings based on weather variation and if weather station variation is above 2.0 that means weather changes frequently and if weather station variation is below 2.0 that means nearly accurate weather.

Questions: Which states have higher accuracy of weather?

Findings: As we can see in our graphs that NT and QLD have higher accuracy of weather as their weather station mean variation below 2.0 most of time.

Questions: Which states have lower accuracy of weather?

Findings: On the other side NSW and VIC have less accuracy of weather as their weather station mean variation above 2.0 most of time.