Avalanche Analysis

Final project analysis of Avalanche data from Snowbound Solutions LLC

Overview of Analysis:

The framework for this project was to analyze avalanche data from Snowbound Solutions LLC based out of Boise, ID and present our findings to the owner, Scott. This data was presented to us from Scott who is a family friend of Rylee's. Observations range from November 2015 to December of 2021 and include different observation locations in Juneau, Alaska with various weather parameters noted as well as a hazard score.

Questions to answer:

We realized early on in our analysis that predicting natural phenomenons are relatively difficult but this analysis might help answer a key question:

What weather features contribute most to Avalanche occurrences in Juneau, Alaska?

Haines
Juneau
Hoonah
Chichagof Island
Admiralty Island

Resources:

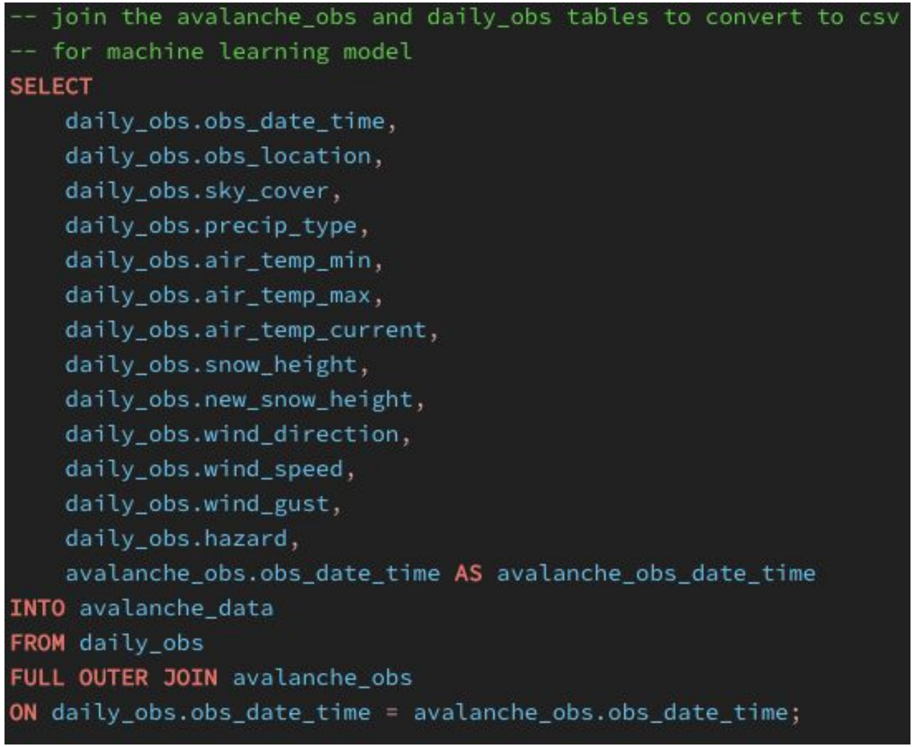
Database: Postgres SQL

Machine Learning: Supervised Model

Coding: Python - Pandas

Visualization: Tableau

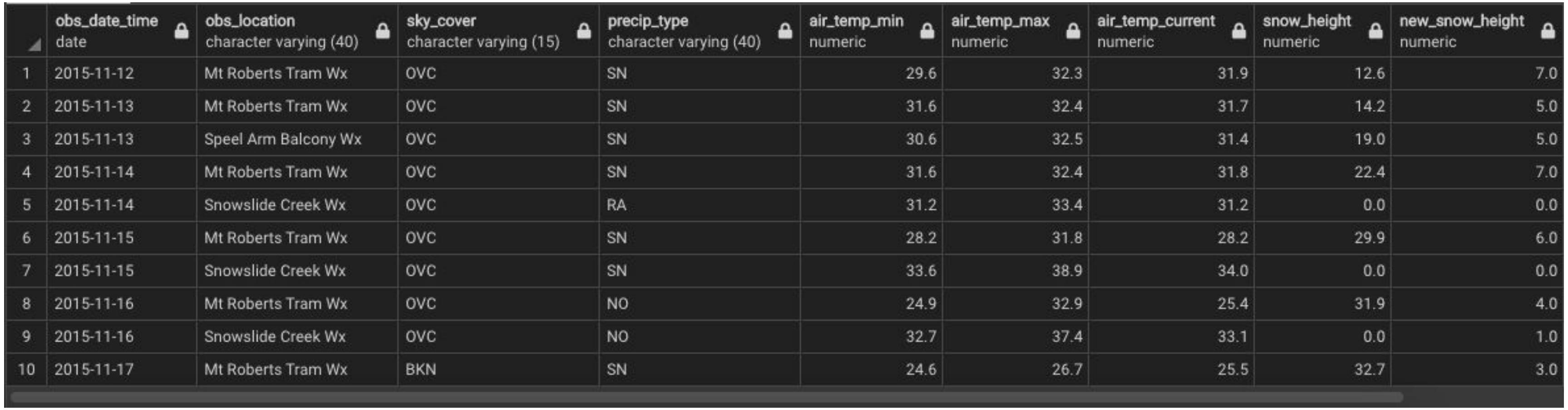
Joined the avalanche_obs and daily_obs provided Json files from Snowbound Solutions LLC into CSV files for our machine learning model



```
-- join the avalanche_obs and daily_obs tables to convert to csv
-- for machine learning model
SELECT
    daily_obs.obs_date_time,
    daily_obs.obs_location,
    daily_obs.sky_cover,
    daily_obs.precip_type,
    daily_obs.air_temp_min,
    daily_obs.air_temp_max,
    daily_obs.air_temp_current,
    daily_obs.snow_height,
    daily_obs.new_snow_height,
    daily_obs.wind_direction,
    daily_obs.wind_speed,
    daily_obs.wind_gust,
    daily_obs.hazard,
    avalanche_obs.obs_date_time AS avalanche_obs_date_time
INTO avalanche_data
FROM daily_obs
FULL OUTER JOIN avalanche_obs
ON daily_obs.obs_date_time = avalanche_obs.obs_date_time;
```

Purpose

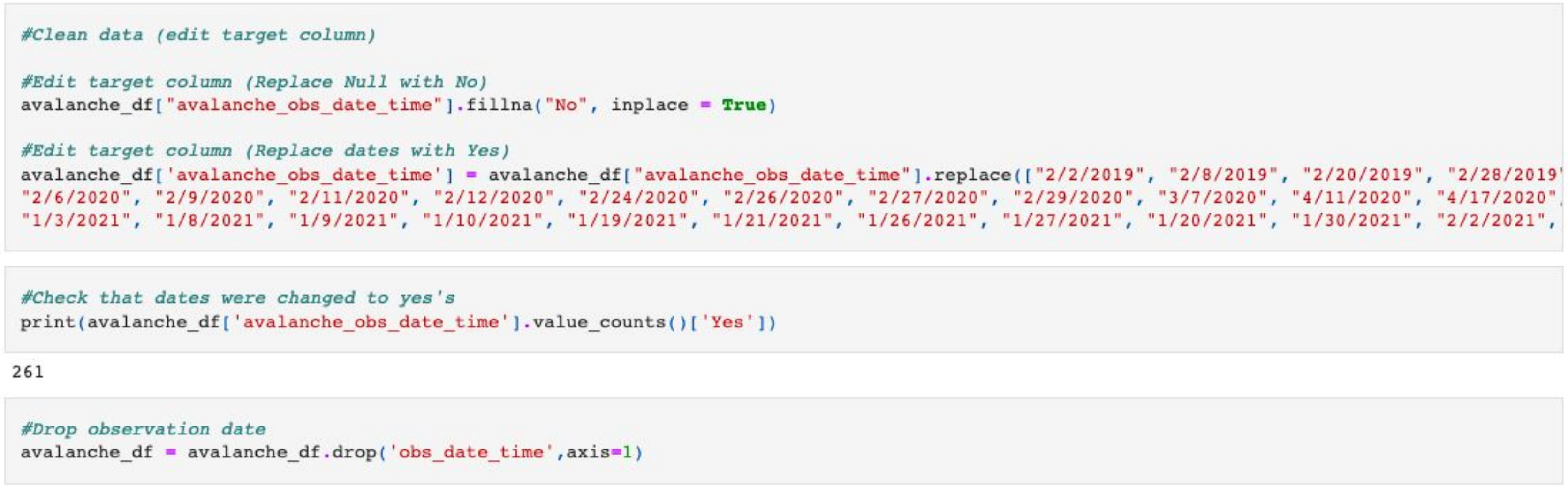
Analysing weather conditions are a critical piece of information for building avalanche forecasts or assessing avalanche hazard for a specific geographic areas. Historically avalanches pose a threat to anyone on snowy mountain sides and can be deadly because of their intensity and seeming unpredictability. By taking the data over the course of several years and multiple areas and examine weather conditions during past avalanches we can predict the probability of an avalanche occurring again based on those factors.



	obs_date_time date	obs_location character varying (40)	sky_cover character varying (15)	precip_type character varying (40)	air_temp_min numeric	air_temp_max numeric	air_temp_current numeric	snow_height numeric	new_snow_height numeric
1	2015-11-12	Mt Roberts Tram Wx	OVC	SN	29.6	32.3	31.9	12.6	7.0
2	2015-11-13	Mt Roberts Tram Wx	OVC	SN	31.6	32.4	31.7	14.2	5.0
3	2015-11-13	Speel Arm Balcony Wx	OVC	SN	30.6	32.5	31.4	19.0	5.0
4	2015-11-14	Mt Roberts Tram Wx	OVC	SN	31.6	32.4	31.8	22.4	7.0
5	2015-11-14	Snowslide Creek Wx	OVC	RA	31.2	33.4	31.2	0.0	0.0
6	2015-11-15	Mt Roberts Tram Wx	OVC	SN	28.2	31.8	28.2	29.9	6.0
7	2015-11-15	Snowslide Creek Wx	OVC	SN	33.6	38.9	34.0	0.0	0.0
8	2015-11-16	Mt Roberts Tram Wx	OVC	NO	24.9	32.9	25.4	31.9	4.0
9	2015-11-16	Snowslide Creek Wx	OVC	NO	32.7	37.4	33.1	0.0	1.0
10	2015-11-17	Mt Roberts Tram Wx	BKN	SN	24.6	26.7	25.5	32.7	3.0

Data Exploration

During the preliminary data preprocessing, we noticed a lot of null values so we dropped those by replacing null values with no as well as replacing the observation dates with yes. Then dropped the observation dates and encoded our categorical columns such as the wind direction, sky cover, precipitation type, etc. Lastly, we scaled the data which is super important when training the model and giving each feature the same footing without any upfront importance.



```
#Clean data (edit target column)

#Edit target column (Replace Null with No)
avalanche_df["avalanche_obs_date_time"].fillna("No", inplace = True)

#Edit target column (Replace dates with Yes)
avalanche_df['avalanche_obs_date_time'] = avalanche_df["avalanche_obs_date_time"].replace(["2/2/2019", "2/8/2019", "2/20/2019", "2/28/2019'
"2/6/2020", "2/9/2020", "2/11/2020", "2/12/2020", "2/24/2020", "2/26/2020", "2/27/2020", "2/29/2020", "3/7/2020", "4/11/2020", "4/17/2020",
"1/3/2021", "1/8/2021", "1/9/2021", "1/10/2021", "1/19/2021", "1/21/2021", "1/26/2021", "1/27/2021", "1/20/2021", "1/30/2021", "2/2/2021",
```

```
#Check that dates were changed to yes's
print(avalanche_df['avalanche_obs_date_time'].value_counts()['Yes'])
```

```
261
```

```
#Drop observation date
avalanche_df = avalanche_df.drop('obs_date_time',axis=1)
```

Data Exploration Continued...

- We decided on logistic regression in our supervised model because we are finding feature importance related to avalanche and hazard level so we kept all of the features since they are all important in this analysis.

- We split the feature and target variables in to X and Y variables, X variable being the feature and Y being the target. Also, we used TRAIN_TEST_SPLIT to split the data in to train and test sets.

- We tried multiple directions but a supervised model helped answer our thesis question the best. A supervised model is the simplest model choice when it comes to optimizing performance criteria using experience and solving various types of real-world computation problems. A benefit specifically to our dataset is that it looks at what features are weighted more heavily and we can clearly look at what features matter by importance. One limitation of this type of model is it is tough to obtain complex relationships

Analysis Phase