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 January 2019 to December of 2021 and include different observation locations in Juneau, Alaska with various weather parameters noted as well as a hazard score.

Question 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?

Resources:

Database: Postgres SQL

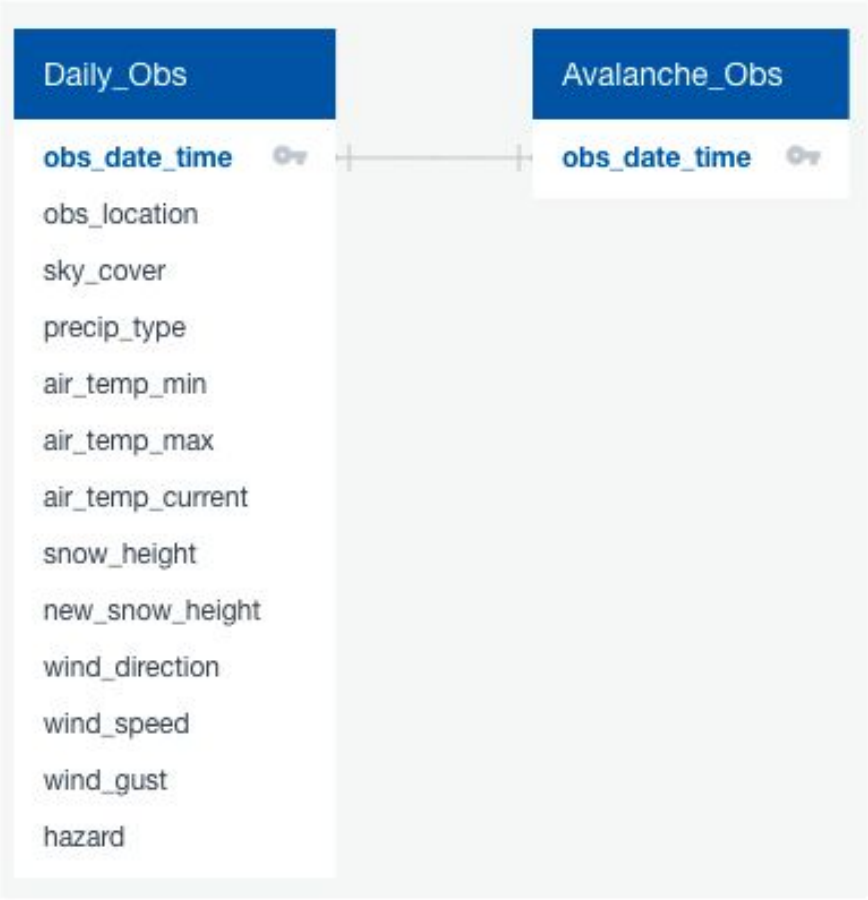
Machine Learning: Supervised Model

Coding: Python - Pandas

Visualization: Tableau

Joined the avalanche_obs and daily_obs files provided by Snowbound Solutions LLC in PGAdmin, then created a connection string using SQLAlchemy to connect our Postgres database to our Jupyter notebook 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
WHERE daily_obs.obs_date_time >= '2019-01-01';
```



Entity relationship diagram connecting the two files (Daily_Obs & Avalanche_Obs)

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 converted the avalanche_occurred column to a yes/no binary column by replacing null values with no as well as replacing the observation dates with yes. Also, dropping null values in the data frame. 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_occured"].fillna("No", inplace = True)

#Edit target column (Replace dates with Yes)
avalanche_df['avalanche_occured'] = avalanche_df["avalanche_occured"].astype(str)
avalanche_df['avalanche_occured'] = avalanche_df["avalanche_occured"].replace(['2019-03-19',
'2020-02-02', '2020-01-14', '2020-02-11', '2020-02-29', '2020-01-31', '2020-02-06',
'2019-02-08', '2019-03-18', '2019-03-03', '2019-02-20', '2020-02-09', '2020-05-01',
'2021-01-26', '2020-02-24', '2021-01-21', '2020-01-15', '2020-01-17', '2021-01-03',
'2021-01-09', '2021-01-08', '2021-01-27', '2021-01-10', '2020-12-25', '2021-01-30',
'2021-02-02', '2020-12-27', '2021-02-09', '2020-04-17', '2020-04-11', '2020-03-07',
'2019-02-02', '2020-02-12', '2020-01-25', '2019-02-28', '2020-11-13', '2020-11-10',
'2021-01-19', '2020-02-26','2020-02-27'], 'Yes')

avalanche_df.head()
```

	obs_date_time	obs_location	sky_cover	precip_type	air_temp_min	air_temp_max	air_temp_current	snow_height	new_snow
0	2019-01-01	Mt Roberts Tram	OVC	RA	-0.1	2.7	2.7	71.9	
1	2019-01-01	Speel Arm Balcony	OVC	RA	0.4	3.0	3.0	52.0	
2	2019-01-01	SS Creek DOT	OVC	RA	3.3	6.9	6.8	0.0	
3	2019-01-01	Snettisham Dorm	OVC	RS	-0.8	0.6	0.3	41.0	
4	2019-01-02	Mt Roberts Tram	OVC	SN	0.1	3.6	0.1	63.0	

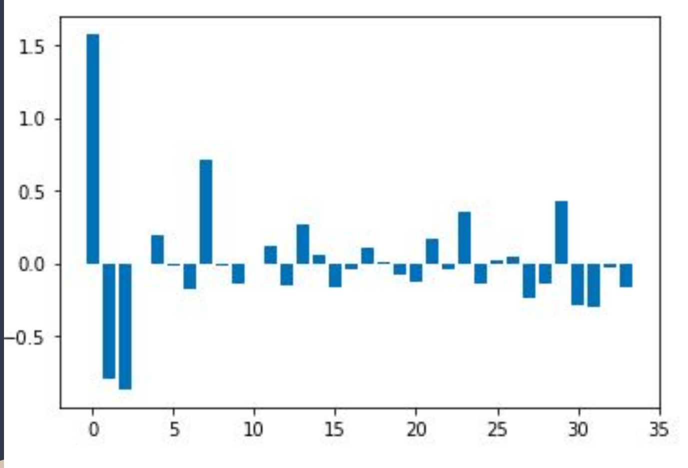
Data Exploration Continued...

- We decided on a feature importance 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 feature importance 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




```
# summarize feature importance
for i,v in enumerate(importance):
    print(f'Feature: %s, Score: %.5f' % (X.columns[i],v))
```

```
Feature: air_temp_min, Score: 1.57297
Feature: air_temp_max, Score: -0.80010
Feature: air_temp_current, Score: -0.86959
Feature: snow_height, Score: -0.00044
Feature: new_snow_height, Score: 0.19357
Feature: wind_speed, Score: -0.02382
Feature: wind_gust, Score: -0.18179
Feature: hazard, Score: 0.71686
Feature: obs_location_Mt Roberts Tram, Score: -0.01335
Feature: obs_location_Mt Roberts Tram Combo Obs, Score: -0.13870
Feature: obs_location_Other, Score: -0.00973
Feature: obs_location_SS Creek DOT, Score: 0.12043
Feature: obs_location_Snettisham Combo Obs, Score: -0.15422
Feature: obs_location_Snettisham Dorm, Score: 0.26204
Feature: obs_location_Speel Arm Balcony, Score: 0.05116
Feature: obs_location_Thane Road Combo Obs, Score: -0.17086
Feature: wind_direction_ESE, Score: -0.03950
Feature: wind_direction_N, Score: 0.11065
Feature: wind_direction_NNE, Score: 0.00364
Feature: wind_direction_NNW, Score: -0.08259
Feature: wind_direction_Other, Score: -0.12926
Feature: wind_direction_SE, Score: 0.16580
Feature: sky_cover_BKN, Score: -0.04187
Feature: sky_cover_CLR, Score: 0.34935
Feature: sky_cover_FEW, Score: -0.14083
Feature: sky_cover_OVC, Score: 0.01368
Feature: sky_cover_SCT, Score: 0.04408
Feature: sky_cover_X, Score: -0.23966
Feature: precip_type_GR, Score: -0.14564
Feature: precip_type_NO, Score: 0.43016
Feature: precip_type_RA, Score: -0.29561
Feature: precip_type_RS, Score: -0.29831
Feature: precip_type_SN, Score: -0.02839
Feature: precip_type_ZR, Score: -0.17012
```