Table of Contents

Impacting the Business with a Distributed Data Science Pipeline
San Diego Street Conditions Classification

Authors: Leonid Shpaner, Jose Luis Estrada, Kiran Singh
Company Name: Street Infrastructure Solutions (SIS)
Company Industry: Streets Infrastructure/Transportation
Company Size: a team of 3 (startup)

Abstract

The city of San Diego has become reliant upon a streets Overall Condition Index (OCI) that was designed and implemented by the United States Army Corps of Engineers. The company will provide recommendations to implement cost savings solutions.

Problem Statement

The city of San Diego has decided to "spend $700,000 to survey the condition of every street in the city so repairs and upgrades can be geared toward increasing social equity, fixing many long-neglected roads and boosting opportunities for bicycling" (Garrick, 2021). The challenge is to identify viable targets (streets) for future infrastructure projects for the city of San Diego. A high caliber consulting service that our company provides is instrumental for handling the following task. Classification of streets in above par conditions is a crucial step in establishing project feasibility. The city's future depends on it.

Goals

1. Predictive Analytics: Predict street viability presence/likelihood (good/fair vs. poor)
2. Prescriptive Analytics: Identify cost effective solution to expand infrastructure projects
3. Informative: Inform City of San Diego of the outcome in a timely manner (by 5/18/2022)

Non-Goals

While we will endeavor to provide recommendations and viable solutions that hinge on sound and proper data analytics, it is not in our capacity to "fix" issues including but not limited to traffic, parking meters, or real-estate assets or valuation.

Data Sources

Data will be stored on AWS service S3 Bucket that will communicate with AWS Sagemaker. The three files will be uploaded to S3 bucket.

- Streets Overall Condition Index (OCI): CSV file 30,712 entries and 12 columns
  https://data.sandiego.gov/datasets/streets-overall-condition-index/

- Street Repair Projects: CSV file with 23,433 entries and 19 columns
  https://data.sandiego.gov/datasets/streets-repair-projects/

- Traffic Volumes: CSV file with 12,432 entries and 10 columns
  https://data.sandiego.gov/datasets/traffic-volumes/

Data Exploration

An S3 bucket is created in which a parent folder directory "raw_files" has three separate folders for each respective CSV file. The data is stored in an S3 Bucket that will communicate with AWS Sagemaker visa vie AWS Athena, a serverless "interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL" (Amazon Web Services, n.d.) to create the database and combine the three files into one single dataframe *df.*

Exploratory Data Analysis (EDA)

During the exploration phase, column names, data types, missing values, and size/shape of the dataset are initially documented in a new cell block. There are a total of 28 columns (features) and 23,005 rows that are a combination of floating point numbers, objects, and integers. Information on whether or not each respective column contains any null or missing values is represented herein. At this stage, missing values are uncovered in date_moratorium (4,426), date_start (1), date_end (7), street_name (16,874), and total_count (16,874), respectively.

Summary Statistics and Outlier Detection

Table 1 shows the summary statistics of the target variable, overall street condition.

Table 1

Overall Condition Index (OCI) Summary

Count	Mean	Std	Min	25%	50%	75%	Max
23,005	74.79141	16.78405	0	66.3	79.06	87.3	100

Note. The mean is lower than the median, suggesting a negatively skewed distribution on the target variable.

Whereas the low (Q1 - 1.5*(IQR)) and high (Q3 + 1.5(IQR)) outliers are found to be 34.8 and 118.8, respectively, omitting these does not benefit long-term project goals. Resistance versus sensitivity to outliers in this endeavor is part and parcel of further analysis.

Data Ingestion

SQL by way of Athena (PyAthena) is used to ingest the data and Pandas is used to read in the SQL queries visa vie the "pd.read_sql()" function. More broadly, the Pandas library is used to read-in and explore the dataframe(s), while matplotlib and seaborn are used for visual explorations (graphical endeavors). An additional helper tool for table visualization (PrettyTable) is leveraged for supplementary visual appeal.

GitHub Repository Information

Main notebook (.ipynb file): https://github.com/lshpaner/sd_streets/blob/main/sd_streets.ipynb
**Main notebook (.pdf file):** https://github.com/lshpaner/sd_streets/blob/main/sd_streets.pdf

Master Link to main repository: https://github.com/lshpaner/sd_streets

Moreover, histograms of all of the numeric features on the joined dataframe are produced to establish or detect the presence of degenerate distributions. One accompanying boxplot examining streets' overall condition index (OCI) is presented visually, but illustrates the same behavior (summary statistics) that has already been depicted in Table 1. Figure 1 shows distributions of all of the numerical features from the entire dataset.

Figure 1

Histogram Distributions
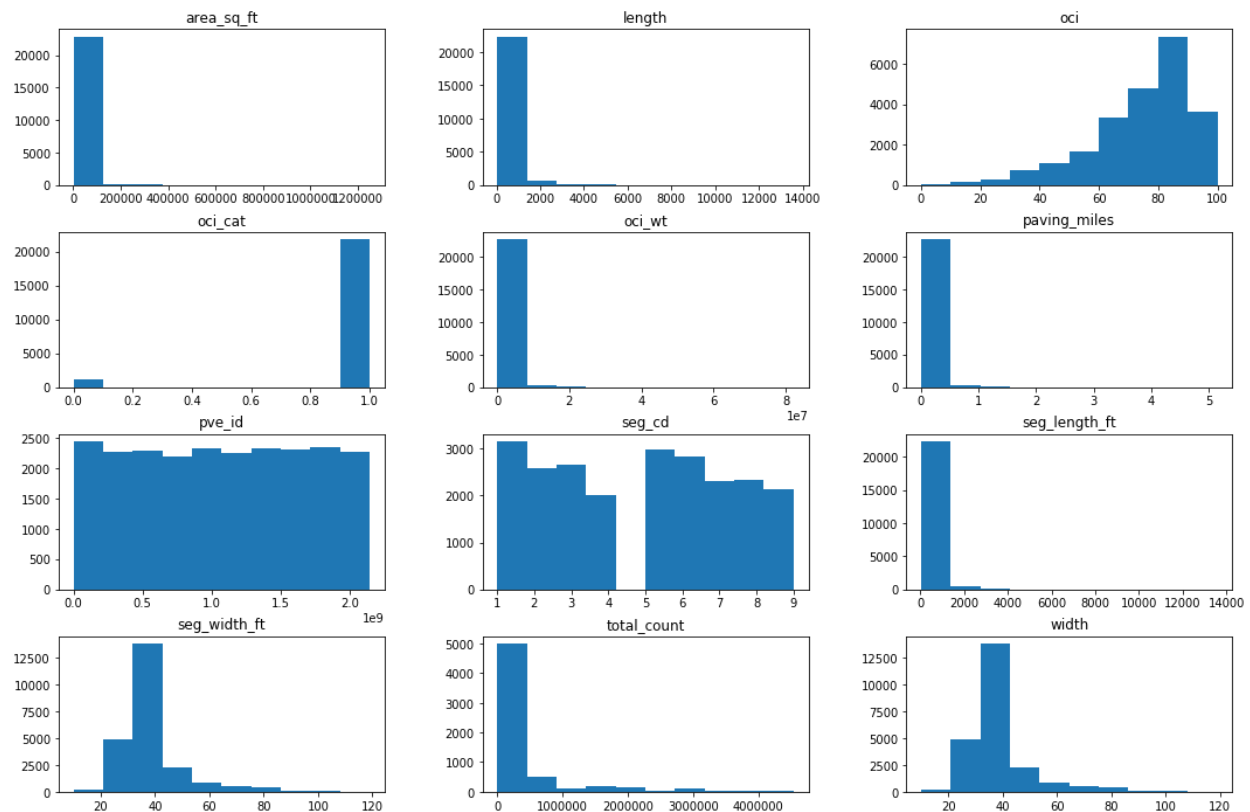
Note. Area in square feet, length, paving miles, and segment length in feet all exhibit right-skewed distributions. The OCI categorical feature is negatively skewed where there is a class imbalance between the 0 and 1 classes, respectively. This is supported by the ensuing Bias Exploration section.

Notwithstanding, all accompanying proportional measurements (i.e., height, width, length, etc.) are true and proper records acquired by the city of San Diego. No immediate transformation (normalization or standardization) is required in order to avoid the potential adverse effect of a high bias, low variance model whereby "a higher bias would not match the data set closely" (Wickramasinghe, 2021).

Pavement identification and total count are of no value and should thus be removed from the dataset.

Bias Exploration

Bias exploration helps determine the extent and/or effect of imbalance data by looking at the target feature of Overall Street Condition Index Description (oci_desc) which provides information on the street quality with "good", "fair," and "poor" conditions, respectively. This effect is measured both numerically and represented visually on a bar graph. There are a total of 23,005 streets of which 6,105 streets are in fair condition, 15,758 streets in good condition, and 1,142 streets in poor condition. Figure 2 shows this categorical distribution in the accompanying bar graph.

Figure 2

Bar Graph of San Diego Street Conditions
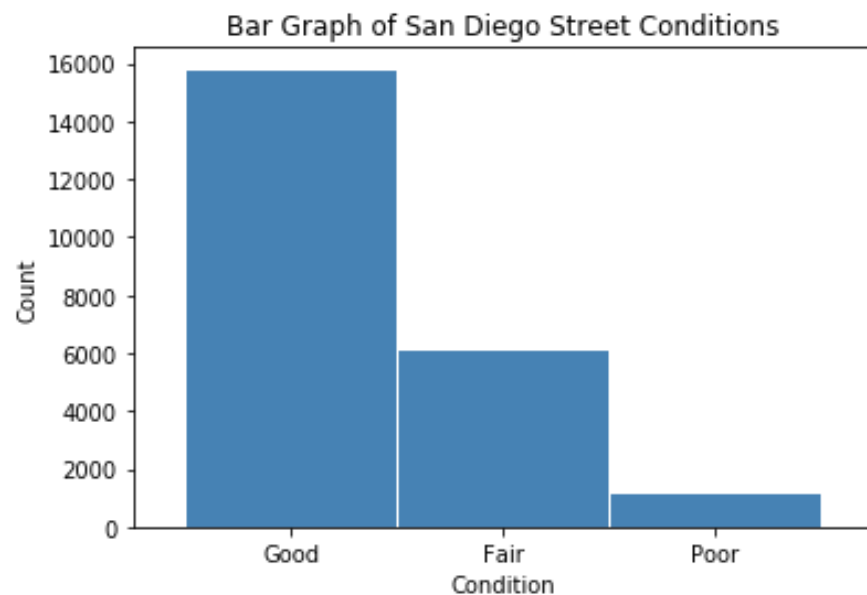
Whereas a method can be used to classify street conditions into multiple classes, it is easier to re-classify streets in "fair" and "good" condition into one category in comparison with the poor class. This, in turn, becomes a binary classification problem. Thus, there are now 21,863 streets in good condition and 1,142 in poor condition (only 5% of all streets). This presents a definitive example of class imbalance.

Class Imbalance

Multiple methods for balancing a dataset exist like "undersampling the majority classes" (Fregly & Barth, 2021, p. 178). To account for the large gap (95%) of mis-classed data on the "poor" condition class, "oversampling the minority class up to the majority class" (p. 179) is commenced. However, such endeavor cannot proceed in good faith without the unsupervised dimensionality reduction technique of Principal Component Analysis (PCA), which is carried out "to compact the dataset and eliminate irrelevant features" (Naseriparsa & Kashani, 2014, p. 33).

In this case, a new dataframe is reduced down into the first two principal components since the largest percent variance explained exists therein and because these principal components are depicted on the ensuing two-dimensional (x,y) scatter plot in Figure 3. A two-dimensional analysis is the most

parsimonious one for illustrating additional visual confirmation of a class imbalance. The two classes are represented by light blue and pinkish red colors, of which the latter captures a larger number of. In other words, streets in good condition encapsulate a larger percentage of data than streets in poor condition.

Figure 3

Class Imbalance in Streets' Overall Condition
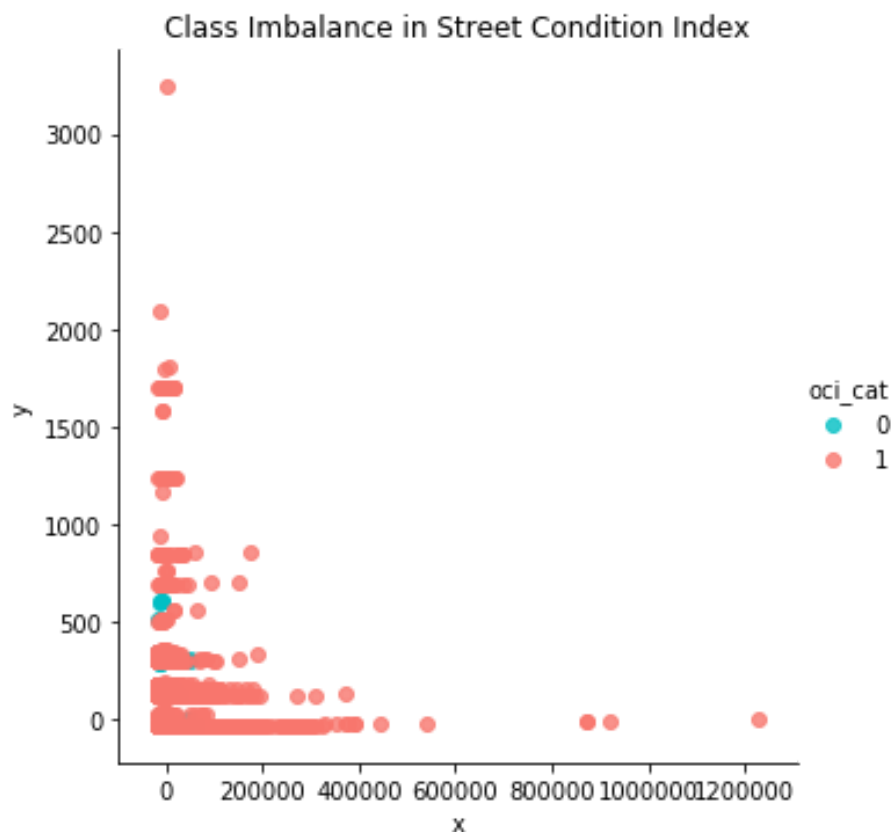
One final exploratory data analysis undertaking yields a triangular correlation matrix, an important step for examining the relationship between predictor variables and determining multicollinearity based on a threshold of a pearson correlation coefficient r = 0.75. Based on this criteria, area in square feet, oci weight, length, width, and paving miles are columns that are earmarked for subsequent removal because length multiplied by width equals to area in square feet. Paving miles is the same information represented on another measurement scale. This is further discussed in the pre-processing section.

Measuring Impact

It is expected that the "good," "fair," and "poor" street condition classes being cast to dummy variables may slightly over-generalize street conditions by placing more emphasis on poorer conditions, specifically within the target variable. To this end, these three variables are narrowed down to two whereby a binary classification follows suit (i.e., good condition vs. poor condition (0,1)). Provided that certain machine learning methods and models have the ability to extract predicted probabilities, this will

allow for a new column with such metrics to be feature engineered at the culmination of predictive modeling.

Security Checklist, Privacy and Other Risks

- No PHI, PII, user behavior, nor credit card data will be stored or processed since the information presented/provided herein is a matter of public record.

- This application will read/write to the following public s3 bucket: s3://waterteam1/raw_files/

- Bias by way of class imbalance is considered and addressed in order to assuage the potential effects of overfitting some or all of the machine learning methods/models that will be explored. Re-balancing the classes where an imbalance exists by oversampling or undersampling is one method of addressing this roadblock.

- One ethical concern that should be addressed is overfitting/underfitting the data commensurate with the initial notions of the viability/efficacy of the dataset at large.

Data Preparation and Data Scrubbing visa vie Pre-Processing

Date_start and date_end are subsequently removed after being concatenated into one uniform feature (date_days). Total count and street name represent the same information, and are unimportant features that are dropped from the dataframe altogether. Moreover, any duplicate columns are removed from the dataframe at large; this is an extra measure for avoiding post-join feature redundancy.

Predictive models only work with numerical values; therefore, categorical features such as func_class (function class), pvm_class (pavement class), and status are transformed into numerical values by mapping dictionaries of categorical values in ascending order. Creation of dummy variables is supported by the following information. For example, in the function class feature, the residential, collector, major, prime, local, and alley functional classes are converted to categorical values (1-6). Similarly, in the pavement class feature, AC Improved, PCC Jointed Concrete, AC Unimproved, and unsurfaced pavement classes are converted to values ranging from one to four.

Lastly, the current status of the job (i.e., post construction, design, bid/award, construction, and planning) is converted to categorical values between one and five. Features with no additional value are removed. Columns with explicit titles (i.e., names) and non-convertible/non-meaningful strings are dropped. Redundant columns (columns that have been cast to dummy variables) are dropped in conjunction with the index column which holds no value in this work.

For context and clarification columns with identifying information that are dropped include the project id, pavement event id, segment id, and project title. Additionally, the project manager's email and phone number are removed to protect sensitive information in accordance with strict compliance standards; moreover, this information cannot be ingested into a viable machine learning algorithm.

Balancing the Dataset

The adaptive synthetic sampling approach (ADAYSN) is used because this is "where more synthetic data is generated for minority class examples that are harder to learn compared to those minority examples that are easier to learn" (He et al., 2008). This allows for the minority class to be more

closely matched (re-sampled) to the majority class for an approximately even 50/50 weight distribution. This results in a larger dataset. Whereas previously there were 23,005 rows, there are now 43,572 (seven rows were previously dropped because of null values).

Train, Test, Validation Splits

To avoid overfitting, the main dataset is split into three respective component parts (dataframes), which will work to train, test, and validate the final model (Solawetz, 2020). Using sci-kit learn, the split is divided by 70%, 15%, and 15%, translating to 30,500 training rows, and 6,536 validation and test rows, respectively. Whereas specifying a stratify parameter allows the split function "to choose any data in the given dataset, causing the splits to become unbalanced" (Fregly & Barth, 2021, p. 181), a random state is set to 777 to ensure reproducible results.

Data Training and Modeling (Classical Approach)

The following experiment has two parts, first of which is conducted using a classical Logistic Regression model-based approach via scikit-learn. Logistic Regression is a viable algorithm for predicting target features that are categorical and with binary outcomes. To this end, the Logistic Regression function is imported directly from the sci-kit learn library, subsequently called, and fitted to the training portion with no hyperparameters other than the random state of 777 for reproducibility. Albeit, discussing performance at this juncture appears to be premature, it is important to note the following. With each iteration of restarting the kernel and running all cells, the accuracy score changes despite a solidified random state on the split data and model itself. Accuracy scores range in a broader-than-expected range of values in the 90th percentile. Modeling data in this way is similar to a "bring-your-own-script" scenario, but it has its limitations in reproducibility. Therefore, likening SageMaker to merely a Jupyter Notebook on the cloud minimizes its efficacy, power, and purpose.

Hyperparameters

To mitigate the classically trained Logistic Regression model, a randomized search is carried out in accordance with a k-fold cross validation of two repeats and five sample splits. A model search space is defined in terms of log uniform values between 1e-5 and 100 and set in conjunction with newton-cg, lbfgs, liblinear solvers, and l1, l2, and elasticnet penalty scoring mechanisms. The number of jobs is set to -1 in order to utilize all of the processing power, parallelizing the cpu usage over 100% of the cores. While accuracy is the predominant scoring criterion in this endeavor, it is only used to establish how often the predictive model is correct given the true positive and true negative scores, divided by the total. If it is higher than the original score, then this score is reproducible provided that the same random state and optimal hyperparameters displayed by the model are used. Precision, recall, and *f1*-scores are still considered by the final output since they are essential performance assessment scores necessary for a thorough, proper, and holistic model evaluation.

Data Training on Refined Algorithm Conducive to a Cloud-Centric Environment

Consequently, SageMaker's built-in XGBoost algorithm is used because it is conducive to sound and proper cloud computing practices within the AWS ecosystem. Moreover, it is an "implementation of gradient boosted decision trees designed for speed and performance" (Brownlee, 2021). To begin the process, an Identity Access Management (IAM) role, region name, and XGboost container are defined. It is important to note that whereas the data has already been split into train, validation, and test sets, this is carried out in accordance with sklearn's classic Logistic Regression model most suitable to an

Anaconda-based coding environment. However, for this rendition of modeling and experimentation, the final dataframe *df2* is once again split, but this time into train (70%) and test (30%) sets only, since model prediction and evaluation has been de-prioritized by the requirements (size and scope) of the project manager. The training data is subsequently transferred into the S3 bucket for storage and retrieval such that it can be called upon for repeated and/or supplemented endeavors.

Parameters

Furthermore, a SageMaker estimator is called upon and passed to an XGBoost variable. Inside the estimator function, the XGBoost container, role, instance count, type, and output path are parsed in.

Instance Size and Count

An instance count is the "number of Amazon EC2 instances to use for training" (Amazon SageMaker Python SDK, n.d.). In this experiment, an instance count of one is used. The instance type, on the other hand, is the "EC2 instance type to deploy this Model to" (SageMaker Python SDK, n.d.). In this case, the instance type is set to *ml.m5.large*, which is justified by a balanced two VPCUs, eight gigabyte memory, and a cost of only $0.115 per hour (AWS, n.d.). This is more than an adequate solution for a project of this scale and magnitude.

Model Evaluation

Whereas various performance metrics exist to assist in the holistic process of model evaluation, an accuracy score alone cannot provide a qualitative assessment of model performance, let alone predictive ability. Regardless, accuracy (~90%) is used in conjunction with precision (86%), recall (94%), and *f1*-score (90%), respectively to shed light on the baseline Logistic Regression performance (first modeling endeavor). These numbers are relatively high, which supports the idea or notion that the model can be used with a different dataset. That being said, since these metrics are not reproducible at a random state of 777, this endeavor does not lend itself to one of repeatability.

Therefore, performance assessment must be relegated to an accuracy score and confusion matrix from SageMaker's built-in XGBoost model. Granted, these are minimal assessments at best, it is discovered that a classification rate of 100% does not provide the necessary framework to move forward with actionable recommendations on street condition classification for the city of San Diego. However, the model has room for improvement and can be re-trained and re-sampled.

Future Enhancements

Enhancement #1: Standardizing/Normalizing The Data

Standardization and normalization techniques are common strategies to improve the performance metrics of predictive models. These techniques are not required on each feature but are very useful when the features have different ranges (Jaitley, 2018). The difference between these two techniques is that normalization rescales the value range from 0 to 1, and standardization rescales the data with a mean equal to zero and a standard deviation of one (Geller, 2019). Additionally, standardization techniques assume a Gaussian distribution in the data, while normalization does not (Lakshmanan, 2019). Therefore, having used these strategies in certain numeric features (i.e., area in square feet and OCI description) could potentially improve the quality of the analysis. However, this remains to be seen in a forthcoming examination of this procedure.

Going back to the pre-processing, the histogram of the feature "area_sq_ft" shows a skewed distribution to the right, meaning that there are outliers present in the data. Eliminating these outliers could negatively impact the modeling since data would be lost. To mitigate the problem, standardization or normalization techniques could be applied to optimize the results of the final model without losing any data.

Enhancement #2: Different Algorithms and Tuning Mechanisms

Though it is clear that choosing a machine learning algorithm suitable for a classification task such as the one presented in this project is a great choice, elaborating on the significance of the choices prior to making them would have potentially benefited the outcome. For example, using XGBoost simply because it was readily available, accessible, and deliverable from a list of SageMaker's built-in algorithms does not consider the need for modeling depth. To this end, understanding the classification problem at hand requires an expanded viewpoint, one that is rich with hyperparameter justifications and clarification on ensemble methods holistically.

It is hereby proposed that the Random Forest model be explored on a subsequent iteration of this project because this model because XGBoost places a larger emphasis on functional space, while the Random Forest classifier "tries to give more preferences to hyperparameters to optimize the model" (Gupta, 2021). When subtracting out the start date from the end date to achieve a total difference in days, it is discovered that there are 18,451 rows with no difference in project duration (zero days), which translates to approximately 80% of the data with immaterial values. Learning from this kind of sparsity would be better if the dataset remained unbalanced for the XGBoost model, but the Random Forest classifier may be a better choice because its hyperparameters are easier to tune and it can "easily adapt to distributed computing than Boosting algorithms" (Gupta, 2021).

Furthermore, an effort can be made to optimize the model with hyperparameters including but not limited to specifying the epochs, which are "the number of passes of the entire training dataset the machine learning algorithm has completed" (Bell, 2020). SageMaker HPT is more than capable of handling this workload since "it supports automatic HPT across multiple algorithms by adding a list of algorithms to the tuning job definition" (Fregly & Barth, 2021, p. 284).

Enhancement #3: Additional Features

Feature selection can impact bias and variance depending on the complexity of the model. Adding new features when the dataset is not big enough will work well on small datasets (El Deeb, 2015). The current dataset has enough observations to increase the number of features. Even if many features were eliminated from the tables, there is an opportunity to explore other datasets that can be merged into the current dataset to improve the overall quality of the final model and recommendations.

The final data set in this analysis consisted of ten features, but additional information such as car and pedestrian traffic can be added to the model. These features are not only helping the modeling part but are adding value to the recommendations that will be provided to the stakeholders. Prioritizing street enhancement by traffic might be in the best interest of the customer to allocate the resources that will impact the population of the city of San Diego. For example, the 24 hour total of all directions of vehicles combined (traffic count) can be reconsidered in the following formula:

$$\text{Traffic Volume} = \text{Total Count} \times \text{Area in Square Feet}$$

To determine the traffic volume, the total count is multiplied by the area in square feet. The 16,874 missing values should be imputed with a value of one such that upon multiplication of the two features, at least the total count would remain in the calculation of volume, whereas imputing with zero would render traffic volume in this new column unreported and immaterial.

Enhancement #4: Infrastructure Amendment Considerations

An S3 bucket on Amazon Web Services has proven to be a viable storage medium for this project. Moreover, future considerations for storage should keep scalability in mind; this simple storage service "is the largest and most performant object storage service for structured and unstructured data and the storage service of choice to build a data lake" (AWS, n.d.). In the same vein, future considerations will be given to scaling up to a data warehouse, with the following advantages in mind. For example, "tables can be organized inside of schemas, which you can think of as folders. When data is ingested, it is stored in various tables described by the schema. Query tools use the schema to determine which data tables to access and analyze" (AWS, n.d.).

References

Amazon Web Services. (n.d.). *Amazon Athena.*
https://aws.amazon.com/athena/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc

Amazon Web Services. (n.d.). *Amazon SageMaker Pricing.* https://aws.amazon.com/sagemaker/pricing/

Amazon Web Services. (n.d.). *Data Lake Storage on AWS.*
https://aws.amazon.com/products/storage/data-lake-storage/?nc=sn&loc=4

Amazon Web Services. (n.d.). *Data Warehouse Concepts.* https://aws.amazon.com/data-warehouse/

Bell, D.J. (2020, May 27). Epoch (machine learning). *Radiopaedia.*
https://radiopaedia.org/articles/epoch-machine-learning?lang=us

Brownlee, J. (2016, August 17). A Gentle Introduction to XGBoost for Applied Machine Learning. *Machine Learning Mastery.*
https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

El Deeb (2015, May 28). 7 Ways to Improve your Predictive Models. *Medium.*
https://medium.com/rants-on-machine-learning/7-ways-to-improve-your-predictive-models-753705eba3d6

Fregly, C. & Barth, A. (2021). *Data Science on AWS.* O'Reilly.

Garrick, D. (2021, September 12). San Diego to spend $700K assessing street conditions to spend repair money wisely. *The San Diego Union-Tribune.*
https://www.sandiegouniontribune.com/news/politics/story/2021-09-12/san-diego-to-spend-700k-assessing-street-conditions-to-spend-repair-money-wisely

Geller, S. (2019, April 4). Normalization vs Standardization – Quantitative analysis. *Towards Data Science.*
https://towardsdatascience.com/normalization-vs-standardization-quantitative-analysis-a91e8a79cebf

Gupta, A. (2021, April 26). XGBoost versus Random Forest. *Medium.*
https://medium.com/geekculture/xgboost-versus-random-forest-898e42870f30

He, H., Bai, Y., Garcia, E. & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence),* 1322-1328. https://ieeexplore.ieee.org/document/4633969

Lakshmanan, S. (2019, May 16). How, When and Why Should You Normalize/Standardize/Rescale Your Data?. *Towards AI.*
https://towardsai.net/p/data-science/how-when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff

Naseriparsa, M. & Kashani, M.M.R. (2014). Combination of PCA with SMOTE Resampling to Boost the Prediction Rate in Lung Cancer Dataset. *International Journal of Computer Applications, 77*(3) 33-38. https://doi.org/10.5120/13376-0987

SageMaker Python SDK. (n.d.). *Estimators.* https://sagemaker.readthedocs.io/en/stable/api/training/estimators.html

Solawetz, J. Train, Validation, Test Split for Machine Learning. *Roboflow.* https://blog.roboflow.com/train-test-split/

Urvashi, J. (2018, October 7). Why Data Normalization is necessary for Machine Learning models. *Medium.* https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029

Wickramasinghe, S. (2021, July 16). Bias & Variance in Machine Learning: Concepts & Tutorials. *Bmc blogs.* https://www.bmc.com/blogs/bias-variance-machine-learning/

Appendix - San Diego Street Conditions Classification

A Cloud Computing Project by Leonid Shpaner, Jose Luis Estrada, and Kiran Singh

```python
[1]: import boto3, re, sys, math, json, os, sagemaker, urllib.request
     import io
     import sagemaker
     from sagemaker import get_execution_role
     from IPython.display import Image
     from IPython.display import display
     from time import gmtime, strftime
     from sagemaker.predictor import csv_serializer
     from pyathena import connect
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     from prettytable import PrettyTable
     from imblearn.over_sampling import SMOTE, ADASYN
     from sklearn.decomposition import PCA
     from sklearn.model_selection import train_test_split, \
     RepeatedStratifiedKFold, RandomizedSearchCV
     from sklearn.metrics import roc_curve, auc, mean_squared_error,\
     precision_score, recall_score, f1_score, accuracy_score,\
     confusion_matrix, plot_confusion_matrix, classification_report
     from sagemaker.tuner import HyperparameterTuner
     from sklearn.linear_model import LogisticRegression
     from sklearn.ensemble import RandomForestClassifier
     from scipy.stats import loguniform
     import warnings
     warnings.filterwarnings('ignore')
```

Data Wrangling

```python
[2]: # create athena database
     sess = sagemaker.Session()
     bucket = sess.default_bucket()
     role = sagemaker.get_execution_role()
     region = boto3.Session().region_name
     # s3 = boto3.Session().client(service_name="s3", region_name=region)

     # ec2 = boto3.Session().client(service_name="ec2", region_name=region)
     # sm = boto3.Session().client(service_name="sagemaker", region_name=region)
```

```python
[3]: ingest_create_athena_db_passed = False
```

```python
[4]: # set a database name
     database_name = "watersd"
```

```python
[5]: # Set S3 staging directory -- this is a temporary directory used for Athena queries
     s3_staging_dir = "s3://{0}/athena/staging".format(bucket)
```

```
[6]: conn = connect(region_name=region, s3_staging_dir=s3_staging_dir)
```

```
[7]: statement = "CREATE DATABASE IF NOT EXISTS {}".format(database_name)
     print(statement)
     pd.read_sql(statement, conn)
```

```
CREATE DATABASE IF NOT EXISTS watersd
```

```
[7]: Empty DataFrame
     Columns: []
     Index: []
```

```
[8]: water_dir = 's3://waterteam1/raw_files'
```

```
[9]: # SQL statement to execute the analyte tests drinking water table

     table_name ='oci_2015_datasd'
     pd.read_sql(f'DROP TABLE IF EXISTS {database_name}.{table_name}', conn)


     create_table = f"""
     CREATE EXTERNAL TABLE IF NOT EXISTS {database_name}.{table_name}(
                 seg_id string,
                 oci float,
                 street string,
                 street_from string,
                 street_to string,
                 seg_length_ft float,
                 seg_width_ft float,
                 func_class string,
                 pvm_class string,
                 area_sq_ft float,
                 oci_desc string,
                 oci_wt float
                 )

                 ROW FORMAT DELIMITED
                 FIELDS TERMINATED BY ','
                 LOCATION '{water_dir}/{table_name}'
                 TBLPROPERTIES ('skip.header.line.count'='1')
     """

     pd.read_sql(create_table, conn)

     pd.read_sql(f'SELECT * FROM {database_name}.{table_name} LIMIT 5', conn)
```

```
[9]:       seg_id    oci street street_from street_to  seg_length_ft  seg_width_ft  \
     0  SA-000003  65.14  ALLEY                              772.7258          30.0
     1  SA-000004  67.45  ALLEY                              196.0025          30.0
     2  SA-000005  70.88  ALLEY                              395.0049          30.0
     3  SA-000006  84.00  ALLEY                              192.0025          30.0
     4  SA-000008  79.24  ALLEY                              251.7540          30.0
```

```
   func_class              pvm_class  area_sq_ft oci_desc      oci_wt
0      Alley  PCC Jointed Concrete   23181.773      Fair  1510060.80
1      Alley  PCC Jointed Concrete    5880.075      Fair   396611.06
2      Alley  PCC Jointed Concrete   11850.147      Good   839938.44
3      Alley  PCC Jointed Concrete    5760.075      Good   483846.30
4      Alley  PCC Jointed Concrete    7552.620      Good   598469.60
```

```python
[10]:  # SQL statement to execute the analyte tests drinking water table

       table_name2 ='sd_paving_datasd'
       pd.read_sql(f'DROP TABLE IF EXISTS {database_name}.{table_name2}', conn)


       create_table = f"""
       CREATE EXTERNAL TABLE IF NOT EXISTS {database_name}.{table_name2}(
                   pve_id int,
                   seg_id string,
                   project_id string,
                   title string,
                   project_manager string,
                   project_manager_phone string,
                   status string,
                   type string,
                   resident_engineer string,
                   address_street string,
                   street_from string,
                   street_to string,
                   seg_cd int,
                   length int,
                   width int,
                   date_moratorium date,
                   date_start date,
                   date_end date,
                   paving_miles float
                   )

                   ROW FORMAT DELIMITED
                   FIELDS TERMINATED BY ','
                   LOCATION '{water_dir}/{table_name2}'
                   TBLPROPERTIES ('skip.header.line.count'='1')
       """

       pd.read_sql(create_table, conn)

       pd.read_sql(f'SELECT * FROM {database_name}.{table_name2} LIMIT 5', conn)
```

```
[10]:         pve_id     seg_id project_id               title  \
       0  1073577074  SA-000319       UTLY  Public Works CIP
       1  1792486183  SA-000345       UTLY  Public Works CIP
       2  1173780646  SA-000375       UTLY  Public Works CIP
```

3

```
3  1276790298  SA-000378       UTLY  Public Works CIP
4    27170959  SA-001081       UTLY  Public Works CIP

         project_manager project_manager_phone           status  \
0  Engineering@sandiego.gov          858-627-3200  post construction
1  Engineering@sandiego.gov          858-627-3200  post construction
2  Engineering@sandiego.gov          858-627-3200  post construction
3  Engineering@sandiego.gov          858-627-3200  post construction
4  Engineering@sandiego.gov          858-627-3200  post construction

      type resident_engineer address_street street_from street_to  seg_cd  \
0   Overlay               ECP          ALLEY                             2
1    Slurry               ECP          ALLEY                             2
2    Slurry               ECP          ALLEY                             2
3    Slurry               ECP          ALLEY                             2
4  Concrete               ECP          ALLEY                             9

   length  width date_moratorium  date_start    date_end  paving_miles
0       0    NaN      2019-02-02  2019-02-02  2019-02-02      0.000000
1     938   30.0      2019-01-30  2019-01-30  2019-01-30      0.177652
2     674   30.0      2018-08-01  2018-08-01  2018-08-01      0.127652
3     658   30.0      2018-08-01  2018-08-01  2018-08-01      0.124621
4     680   30.0            None  2020-08-13  2020-08-13      0.128788
```

```python
[11]: # SQL statement to execute the analyte tests drinking water table

table_name3 ='traffic_counts_datasd'
pd.read_sql(f'DROP TABLE IF EXISTS {database_name}.{table_name3}', conn)


create_table = f"""
CREATE EXTERNAL TABLE IF NOT EXISTS {database_name}.{table_name3}(
            id string,
            street_name string,
            limits string,
            northbound_count int,
            southbound_count int,
            eastbound_count int,
            westbound_count int,
            total_count int,
            file_no string,
            date_count date
            )

            ROW FORMAT DELIMITED
            FIELDS TERMINATED BY ','
            LOCATION '{water_dir}/{table_name3}'
            TBLPROPERTIES ('skip.header.line.count'='1')
"""

pd.read_sql(create_table, conn)
```

```python
pd.read_sql(f'SELECT * FROM {database_name}.{table_name3} LIMIT 5', conn)
```

[11]:
	id	street_name	limits	northbound_count \
0	01AV018207	01 AV	A ST - ASH ST	18010
1	01AV015210	01 AV	A ST - ASH ST	20060
2	01AV018213	01 AV	A ST - ASH ST	19597
3	01AV007721	01 AV	A ST - ASH ST	10640
4	01AV088812	01 AV	ASH ST - BEECH ST	2298

	southbound_count	eastbound_count	westbound_count	total_count	file_no \
0	None	None	None	18010	0182-07
1	None	None	None	20060	0152-10
2	None	None	None	19597	0182-13
3	None	None	None	10640	0077-21
4	None	None	None	2298	0888-12

	date_count
0	2007-03-13
1	2010-03-18
2	2013-03-12
3	2021-03-10
4	2012-12-11

[12]:
```python
statement = "SHOW DATABASES"
df_show = pd.read_sql(statement, conn)
df_show.head(5)
```

[12]:
	database_name
0	default
1	dsoaws
2	watersd

[13]:
```python
if database_name in df_show.values:
    ingest_create_athena_db_passed = True
```

[14]:
```python
%store ingest_create_athena_db_passed
```

Stored 'ingest_create_athena_db_passed' (bool)

[15]:
```python
pd.read_sql(f'SELECT * FROM {database_name}.{table_name} t1 INNER JOIN \
                          {database_name}.{table_name2} t2 ON t1.seg_id \
                          = t2.seg_id LIMIT 5', conn)
```

[15]:
	seg_id	oci	street	street_from	street_to	seg_length_ft \
0	SA-000345	34.14	ALLEY			937.9261
1	SA-000375	97.25	ALLEY			673.3209
2	SA-000378	62.67	ALLEY			657.2000
3	SA-001081	68.86	ALLEY			679.1060
4	SA-001083	28.67	ALLEY			660.0917

	seg_width_ft	func_class	pvm_class	area_sq_ft	… \

```
0          30.0        Alley            AC Improved   28137.783  …
1          30.0        Alley  PCC Jointed Concrete   20199.627  …
2          30.0        Alley  PCC Jointed Concrete   19716.000  …
3          30.0        Alley  PCC Jointed Concrete   20373.180  …
4          30.0        Alley  PCC Jointed Concrete   19802.752  …

   address_street  street_from  street_to seg_cd length width date_moratorium  \
0          ALLEY                                2    938    30      2019-01-30
1          ALLEY                                2    674    30      2018-08-01
2          ALLEY                                2    658    30      2018-08-01
3          ALLEY                                9    680    30            None
4          ALLEY                                9    661    30            None

   date_start    date_end paving_miles
0  2019-01-30  2019-01-30     0.177652
1  2018-08-01  2018-08-01     0.127652
2  2018-08-01  2018-08-01     0.124621
3  2020-08-13  2020-08-13     0.128788
4  2020-07-31  2020-07-31     0.125189

[5 rows x 31 columns]
```

```python
[16]: df = pd.read_sql(f'SELECT * FROM (SELECT * FROM {database_name}.{table_name} \
                       t1 INNER JOIN {database_name}.{table_name2} t2 \
                       ON t1.seg_id = t2.seg_id) m1 LEFT JOIN (SELECT street_name,␣
  ↪\
                                                  SUM(total_count)␣
  ↪total_count \
                                                  FROM␣
  ↪{database_name}.{table_name3} \
                                                  GROUP BY␣
  ↪street_name) t3 \
                       ON m1.address_street = t3.street_name', conn)
```

```python
[17]: df.head(5)
```

```
[17]:       seg_id    oci street street_from street_to  seg_length_ft  seg_width_ft  \
      0  SA-000345  34.14  ALLEY                              937.9261          30.0
      1  SA-000375  97.25  ALLEY                              673.3209          30.0
      2  SA-000378  62.67  ALLEY                              657.2000          30.0
      3  SA-001081  68.86  ALLEY                              679.1060          30.0
      4  SA-001083  28.67  ALLEY                              660.0917          30.0

   func_class            pvm_class  area_sq_ft  … street_to  seg_cd  length  \
0      Alley            AC Improved   28137.783  …                 2     938
1      Alley  PCC Jointed Concrete   20199.627  …                 2     674
2      Alley  PCC Jointed Concrete   19716.000  …                 2     658
3      Alley  PCC Jointed Concrete   20373.180  …                 9     680
4      Alley  PCC Jointed Concrete   19802.752  …                 9     661

   width date_moratorium  date_start    date_end paving_miles street_name  \
```

```
0      30       2019-01-30   2019-01-30   2019-01-30      0.177652          None
1      30       2018-08-01   2018-08-01   2018-08-01      0.127652          None
2      30       2018-08-01   2018-08-01   2018-08-01      0.124621          None
3      30             None   2020-08-13   2020-08-13      0.128788          None
4      30             None   2020-07-31   2020-07-31      0.125189          None

   total_count
0          NaN
1          NaN
2          NaN
3          NaN
4          NaN

[5 rows x 33 columns]
```

```python
[18]:  # remove duplicated columns
       df = df.loc[:,~df.columns.duplicated()]
```

```python
[19]:  # create flat .csv file from originally
       # merged dataframe
       # df.to_csv('original_merge.csv')
```

Exploratory Data Analysis (EDA)

```python
[20]:  # get number of rows and columns
       print('Number of Rows:', df.shape[0])
       print('Number of Columns:', df.shape[1], '\n')

       # inspect datatypes and nulls
       data_types = df.dtypes
       data_types = pd.DataFrame(data_types)
       data_types = data_types.assign(Null_Values =
                                   df.isnull().sum())
       data_types.reset_index(inplace = True)
       data_types.rename(columns={0:'Data Type',
                            'index': 'Column/Variable',
                            'Null_Values': "# of Nulls"})
```

```
Number of Rows: 23005
Number of Columns: 30
```

```
[20]:         Column/Variable Data Type  # of Nulls
       0              seg_id    object           0
       1                 oci   float64           0
       2              street    object           0
       3         street_from    object           0
       4           street_to    object           0
       5       seg_length_ft   float64           0
       6        seg_width_ft   float64           0
       7           func_class    object           0
       8            pvm_class    object           0
```

```
9            area_sq_ft   float64          0
10              oci_desc    object          0
11                oci_wt   float64          0
12                pve_id     int64          0
13            project_id    object          0
14                 title    object          0
15        project_manager    object          0
16  project_manager_phone    object          0
17                status    object          0
18                  type    object          0
19      resident_engineer    object          0
20         address_street    object          0
21                seg_cd     int64          0
22                length     int64          0
23                 width     int64          0
24        date_moratorium    object       4426
25            date_start    object          1
26              date_end    object          7
27          paving_miles   float64          0
28            street_name    object      16874
29            total_count   float64      16874
```

Bias Exploration

To explore potential areas of bias, we will endeavor to trace class imbalance on the target feature of "oci_desc."

```python
[21]: oci_desc_fair = df['oci_desc'].value_counts()['Fair']
      oci_desc_good = df['oci_desc'].value_counts()['Good']
      oci_desc_poor = df['oci_desc'].value_counts()['Poor']
      oci_desc_total = oci_desc_fair  + oci_desc_good + oci_desc_poor

      table1 = PrettyTable() # build a table
      table1.field_names = ['Fair Condition', 'Good Condition',
                            'Poor Condition', 'Total']
      table1.add_row([oci_desc_fair, oci_desc_good, oci_desc_poor,
                    oci_desc_total])
      table1
```

```
[21]: +----------------+----------------+----------------+-------+
      | Fair Condition | Good Condition | Poor Condition | Total |
      +----------------+----------------+----------------+-------+
      |      6105      |     15758      |      1142      | 23005 |
      +----------------+----------------+----------------+-------+
```

```python
[22]: perc_good = oci_desc_good /(oci_desc_total)
      perc_fair = oci_desc_fair /(oci_desc_total)
      perc_poor = oci_desc_poor /(oci_desc_total)
      print(round(perc_good, 2)*100, '% of streets '
                            'are in good condition ')
      print(round(perc_fair, 2)*100, '% of streets '
                            'are in fair condition ')
```

```
print(round(perc_poor, 2)*100, '% of streets '
                         'are in poor condition ')
```

```
68.0 % of streets are in good condition
27.0 % of streets are in fair condition
5.0 % of streets are in poor condition
```

Considerably more than half of the streets are in good condition. A little less than a third are in fair condition. Only 5% are in poor condition.

[23]:
```
# accidents injury bar graph
conditions = df['oci_desc'].value_counts()
fig = plt.figure()
conditions.plot.bar(x ='lab', y='val', rot=0, width=0.99,
                         color="steelblue")
plt.title ('Bar Graph of San Diego Street Conditions')
plt.xlabel('Condition')
plt.ylabel('Count')
plt.show()

conditions
```
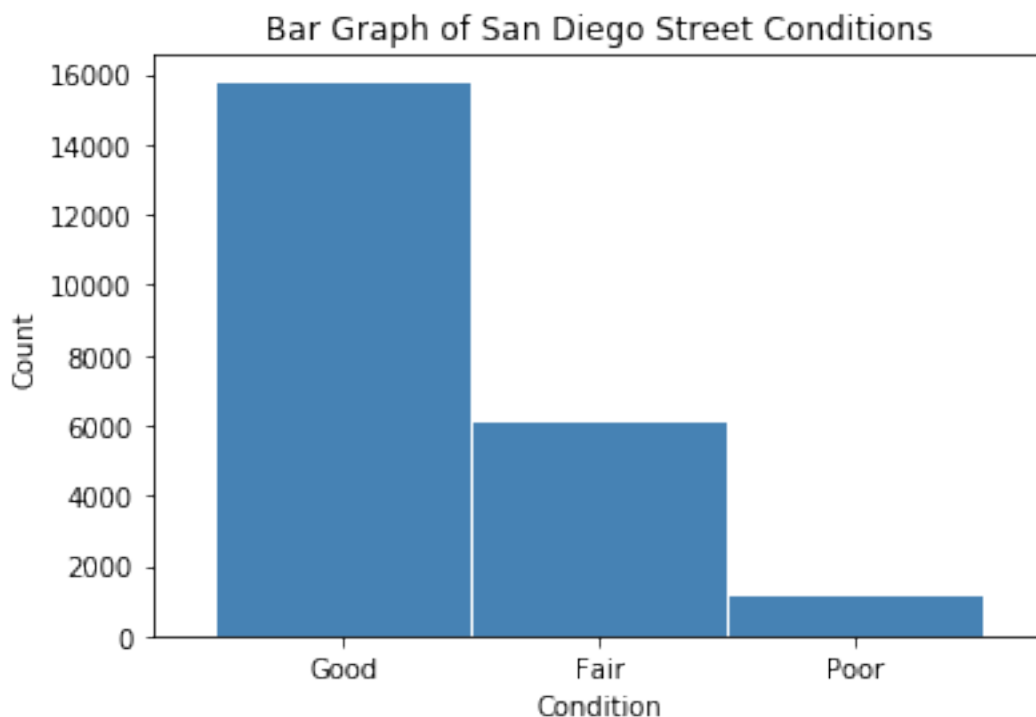

[23]:
```
Good     15758
Fair      6105
Poor      1142
Name: oci_desc, dtype: int64
```

Whereas a method can be used to classify street conditions into multiple classes, it is easier to re-classify streets in "fair" and "good" condition into one category in comparison with the poor class. This, in turn,

9

becomes a binary classification problem. Thus, there are now 21,863 streets in good condition and 1,142 in poor condition (only 5% of all streets). This presents a definitive example of class imbalance.
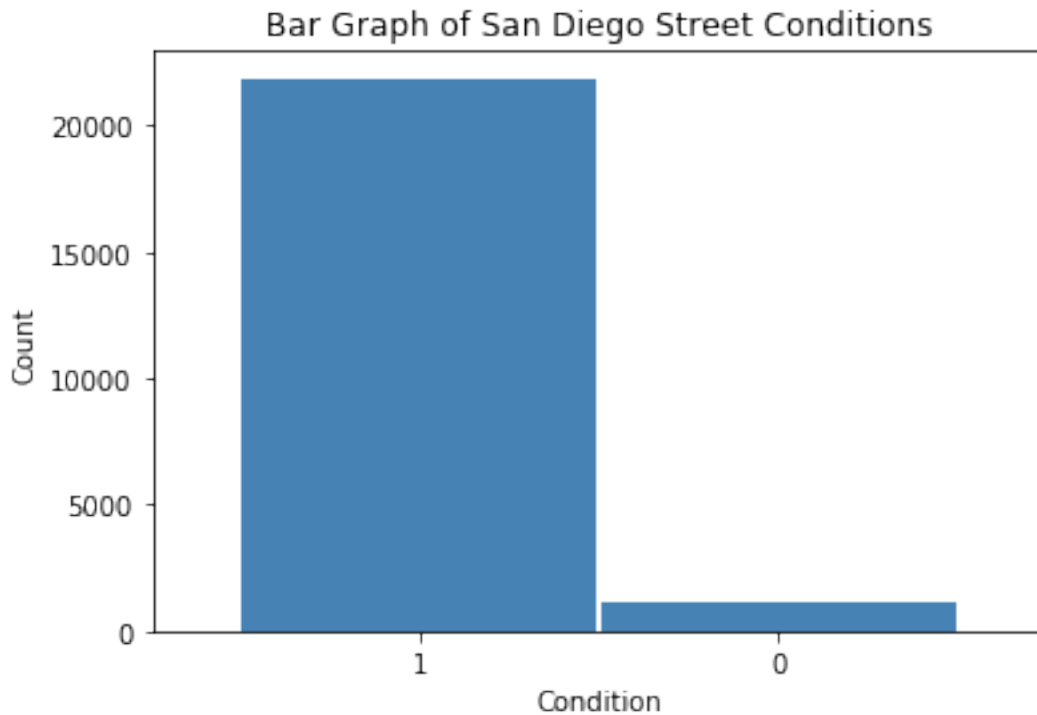
```
[24]: df['oci_cat'] = df['oci_desc'].map({'Good':1, 'Fair':1,
                                          'Poor':0})
      cond = df['oci_cat'].value_counts()
      cond
```

```
[24]: 1    21863
      0     1142
      Name: oci_cat, dtype: int64
```

```
[25]: # oci ratings bar graph
      fig = plt.figure()
      cond.plot.bar(x ='lab', y='val', rot=0, width=0.99,
                                color="steelblue")
      plt.title ('Bar Graph of San Diego Street Conditions')
      plt.xlabel('Condition')
      plt.ylabel('Count')
      plt.show()

      cond
```



```
[25]: 1    21863
      0     1142
      Name: oci_cat, dtype: int64
```

```
[26]:  # cast oci info into range of values
       labels = [ "{0} - {1}".format(i, i + 5) for i in range(0, 100, 10) ]
       df['OCI Range'] = pd.cut(df.oci, range(0, 105, 10),
                                       right=False,
                                       labels=labels).astype(object)
       # inspect the new dataframe with this info
       df[['oci', 'OCI Range']]
```

```
[26]:          oci OCI Range
       0      34.14   30 - 35
       1      97.25   90 - 95
       2      62.67   60 - 65
       3      68.86   60 - 65
       4      28.67   20 - 25
       ...      ...       ...
       23000  93.40   90 - 95
       23001  91.01   90 - 95
       23002  97.26   90 - 95
       23003  95.00   90 - 95
       23004  80.83   80 - 85

       [23005 rows x 2 columns]
```

```
[27]:  print("\033[1m"+'Street Conditions by Condition Index Range'+"\033[1m")
       def oci_cond():
           oci_desc_good = df.loc[df.oci_desc == 'Good'].groupby(
                              ['OCI Range'])[['oci_desc']].count()
           oci_desc_good.rename(columns = {'oci_desc':'Good'}, inplace=True)
           oci_desc_fair = df.loc[df.oci_desc == 'Fair'].groupby(
                              ['OCI Range'])[['oci_desc']].count()
           oci_desc_fair.rename(columns = {'oci_desc':'Fair'}, inplace=True)
           oci_desc_poor = df.loc[df.oci_desc == 'Poor'].groupby(
                              ['OCI Range'])[['oci_desc']].count()
           oci_desc_poor.rename(columns = {'oci_desc':'Poor'}, inplace=True)
           oci_desc_comb = pd.concat([oci_desc_good, oci_desc_fair, oci_desc_poor],
           axis = 1)
           # sum row totals
           oci_desc_comb.loc['Total']= oci_desc_comb.sum(numeric_only=True, axis=0)
           # sum column totals
           oci_desc_comb.loc[:,'Total'] = oci_desc_comb.sum(numeric_only=True, axis=1)
           oci_desc_comb.fillna(0, inplace = True)
           return oci_desc_comb.style.format("{:,.0f}")

       oci_cond = oci_cond().data # retrieve dataframe
       oci_cond
```

       Street Conditions by Condition Index Range

[27]:   Good   Fair   Poor   Total
       70 - 75 4766.0   3.0   0.0   4769.0
       80 - 85 7341.0   0.0   0.0   7341.0
       90 - 95 3541.0   0.0   0.0   3541.0

```
40 - 45        0.0  1095.0      0.0   1095.0
50 - 55        0.0  1685.0      0.0   1685.0
60 - 65        0.0  3322.0      0.0   3322.0
0 - 5          0.0     0.0     37.0     37.0
10 - 15        0.0     0.0    135.0    135.0
20 - 25        0.0     0.0    259.0    259.0
30 - 35        0.0     0.0    711.0    711.0
Total     15648.0  6105.0   1142.0  22895.0
```

```python
[28]: oci_plt = oci_cond['Total'][0:8].sort_values(ascending=False)
      oci_plt.plot(kind='bar', width=0.90)
      plt.title('Street Conditions by Index Range')
      plt.xlabel('Index Range')
      plt.ylabel('# of Streets')
      plt.show()
```
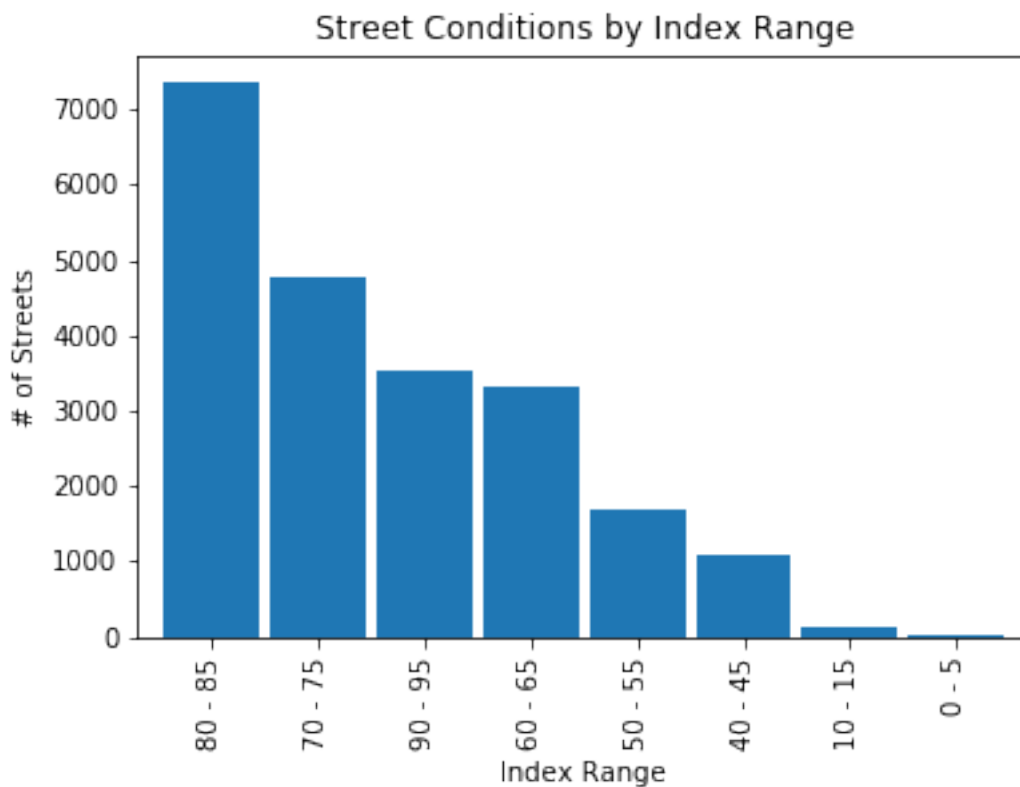

Summary Statistics

```python
[29]: # summary statistics
      summ_stats = pd.DataFrame(df['oci'].describe()).T
      summ_stats
```

```
[29]:        count       mean        std  min   25%    50%   75%    max
      oci  23005.0  74.791413  16.784048  0.0  66.3  79.06  87.3  100.0
```

```
[30]: IQR = summ_stats['75%'][0] - summ_stats['25%'][0]
      low_outlier = summ_stats['25%'][0] - 1.5*(IQR)
      high_outlier = summ_stats['75%'][0] + 1.5*(IQR)

      print('Low Outlier:', low_outlier)
      print('High Outlier:', high_outlier)
```

Low Outlier: 34.8
High Outlier: 118.8

```
[31]: print("\033[1m"+'Overall Condition Index (OCI) Summary'+"\033[1m")
      def oci_by_range():
          pd.options.display.float_format = '{:,.2f}'.format
          new = df.groupby('OCI Range')['oci']\
          .agg(["mean",
                "median",
                "std",
                "min",
                "max"])

          new.loc['Total'] = new.sum(numeric_only=True, axis=0)
          column_rename = {'mean': 'Mean', 'median': 'Median',
                           'std': 'Standard Deviation',\
                           'min':'Minimum','max': 'Maximum'}

          dfsummary = new.rename(columns = column_rename)
          return dfsummary

      oci_by_range = oci_by_range()
      oci_by_range
```
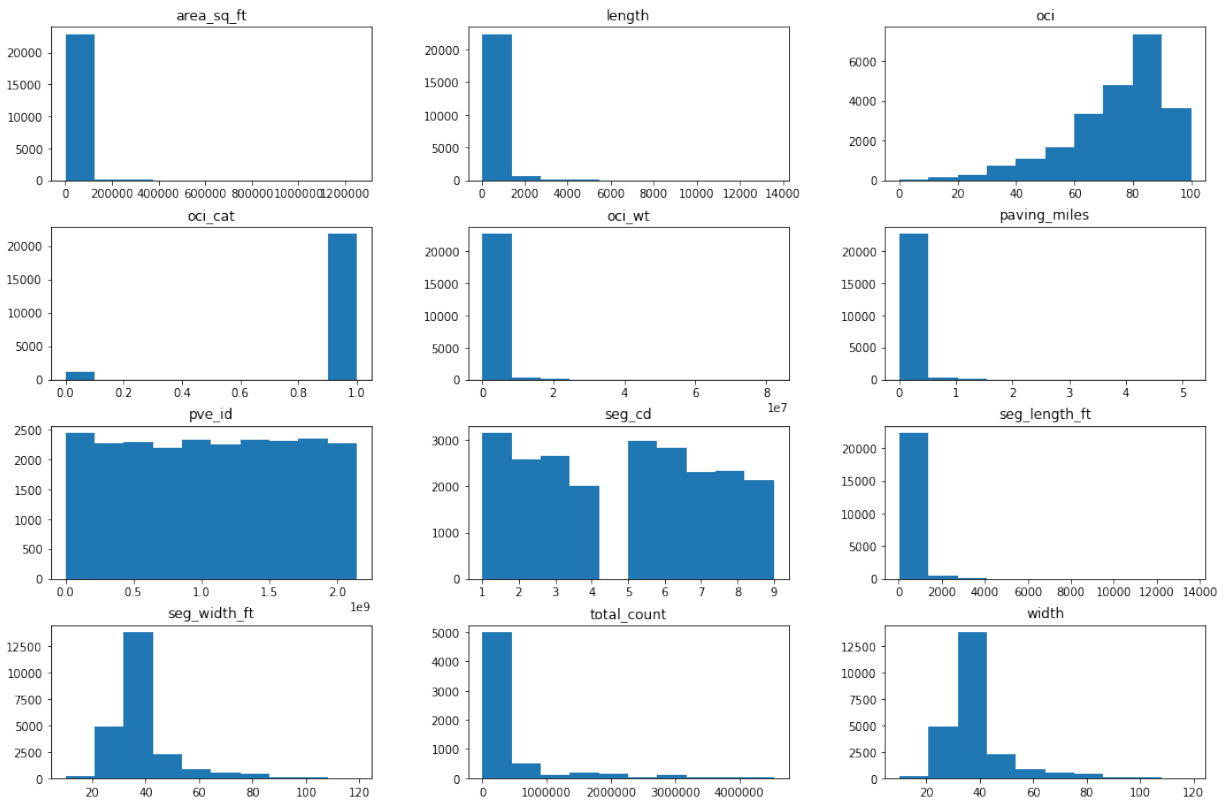
Overall Condition Index (OCI) Summary

[31]:

	Mean	Median	Standard Deviation	Minimum	Maximum
OCI Range					
0 - 5	6.13	8.00	3.70	0.00	9.69
10 - 15	15.66	16.40	2.82	10.11	19.84
20 - 25	25.77	26.17	2.91	20.12	29.96
30 - 35	35.63	36.04	2.80	30.04	39.98
40 - 45	45.37	45.58	2.88	40.00	49.98
50 - 55	55.62	56.00	2.88	50.00	59.98
60 - 65	65.56	65.80	2.82	60.00	69.99
70 - 75	75.11	75.16	2.97	70.00	79.99
80 - 85	85.14	85.15	2.84	80.00	89.99
90 - 95	93.44	92.89	2.57	90.00	99.33
Total	503.42	507.19	29.18	450.27	548.73

Histogram Distributions

```
[32]: # histograms
      df.hist(grid=False, figsize=(18,12))
      plt.show()
```

Boxplot Distribution (OCI)

```
[33]:  # selected boxplot distribution for oci values
       print("\033[1m"+'Boxplot Distribution'+"\033[1m")

       # Boxplot of age as one way of showing distribution
       fig = plt.figure(figsize = (10,1.5))
       plt.title ('Boxplot: Overall Condition Index (OCI)')
       plt.xlabel('Speed Limit')
       plt.ylabel('Value')

       sns.boxplot(data=df['oci'],
                   palette="coolwarm",
                   orient='h',
                   linewidth=2.5)
       plt.show()

       IQR = summ_stats['75%'][0] - summ_stats['25%'][0]

       print('The first quartile is %s. '%summ_stats['25%'][0])
       print('The third quartile is %s. '%summ_stats['75%'][0])
       print('The IQR is %s.'%round(IQR,2))
       print('The mean is %s. '%round(summ_stats['mean'][0],2))
       print('The standard deviation is %s. '%round(summ_stats['std'][0],2))
       print('The median is %s. '%round(summ_stats['50%'][0],2))
```
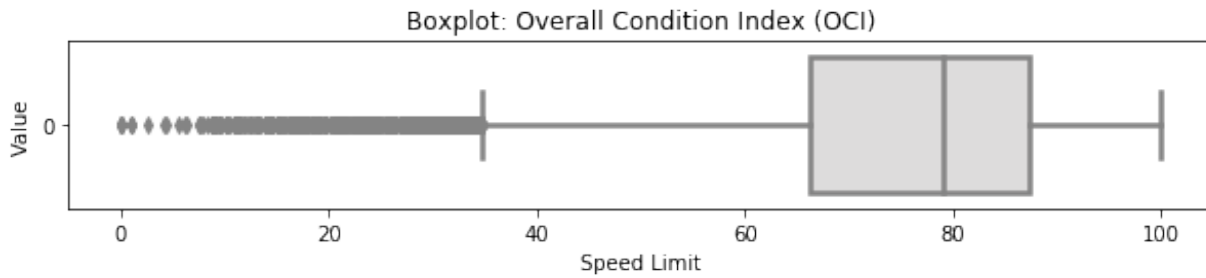
14

Boxplot Distribution

Boxplot: Overall Condition Index (OCI)

The first quartile is 66.3.
The third quartile is 87.3.
The IQR is 21.0.
The mean is 74.79.
The standard deviation is 16.78.
The median is 79.06.

Correlation Matrix

```python
[34]:  # assign correlation function to new variable

       corr = df.corr()

       # for triangular matrix

       matrix = np.triu(corr)

       plt.figure(figsize=(
                       10,10
                        )
                        )

       # parse corr variable into triangular matrix

       sns.heatmap(df.corr(
                       method='pearson'),
                       annot=True,
                       linewidths=.5,
                       cmap="coolwarm",
                       mask=matrix,
                       square = True,
                       cbar_kws={'label': 'Correlation Index'},
                       vmin=-1,
                       vmax=1
                        )

       plt.show()
```
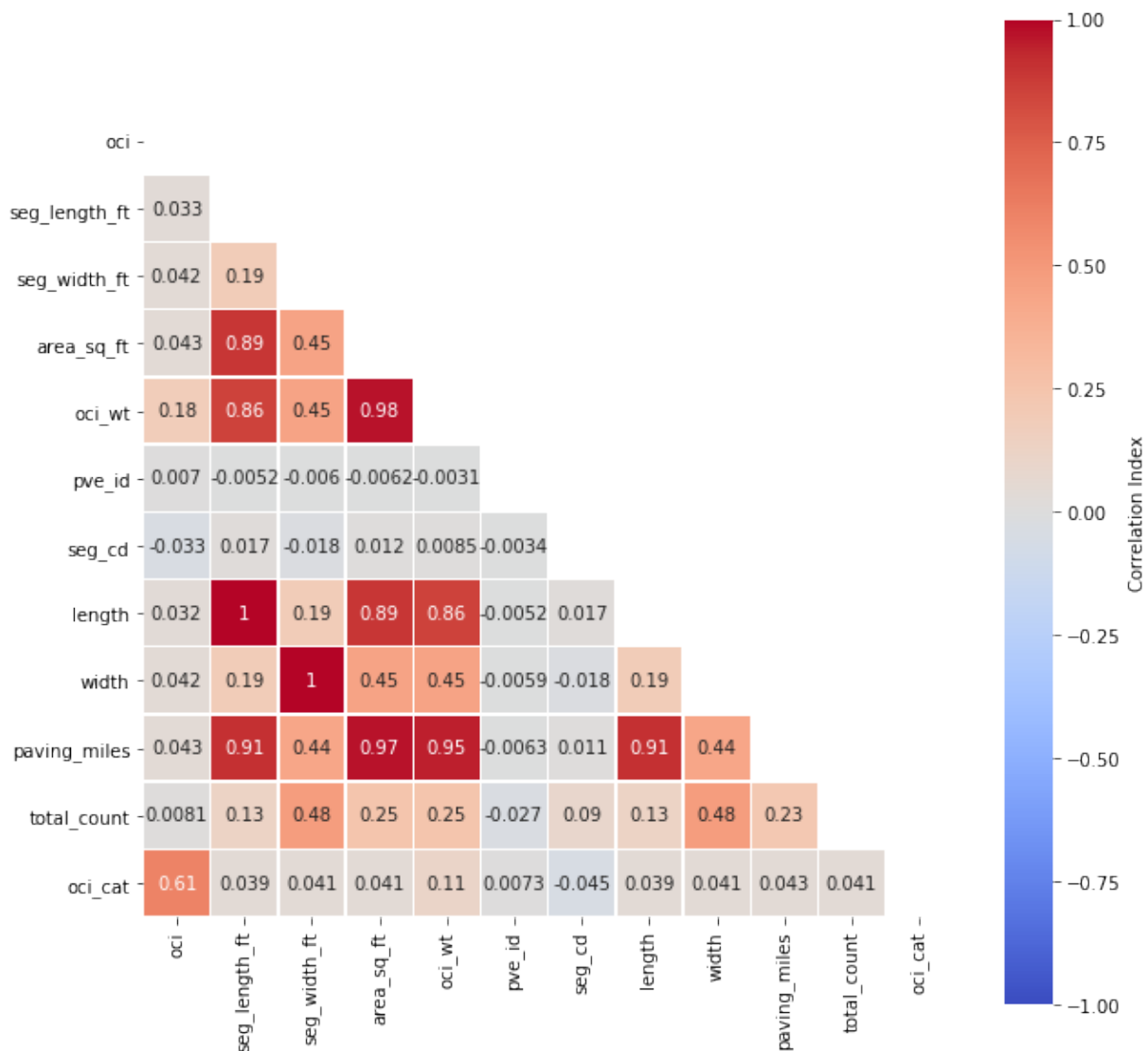
Multicollinearity

Let us narrow our focus by removing highly correlated predictors and passing the rest into a new dataframe.

```
[35]: cor_matrix = df.corr().abs()

      upper_tri = cor_matrix.where(np.triu(np.ones(cor_matrix.shape),
                                           k=1).astype(np.bool))

      to_drop = [column for column in upper_tri.columns if
                 any(upper_tri[column] > 0.75)]

      print('These are the columns prescribed to be dropped: %s'%to_drop)
```

These are the columns prescribed to be dropped: ['area_sq_ft', 'oci_wt', 'length',
'width', 'paving_miles']

Pre-Processing

Based on the prescribed output of the multicollinearity outcome, we should remove `area_sq_ft`, `oci_wt`, `length`, `width`, `paving_miles`, respectively. However, area in square feet is derived from length (x) width values, and converted to paving miles. Removing all of these features is not necessary. We can keep area in square feet, as long as we remove the rest.

Feature Engineering

The start date is subtracted from the end date and converted to number of days as one column.

```python
[36]: df['date_end'] = pd.to_datetime(df['date_end'])
      df['date_start'] = pd.to_datetime(df['date_start'])

      # 7 rows with missing values are dropped in the following line
      day_diff = df.dropna(subset=['date_end',
                                   'date_start'],
                           inplace=True)

      df['day_diff'] = (df['date_end'] - df['date_start']).dt.days.astype(int)
```

```python
[37]: zero_days = df['day_diff'].value_counts()[0]
      percent_days = round(zero_days/len(df), 2)*100
      print('There are', zero_days, 'rows with "0".')
      print('That is roughly', percent_days, '% of the data.')
```

There are 18451 rows with "0".
That is roughly 80.0 % of the data.

The residential, collector, major, prime, local, and alley functional classes are converted to dummy variables.

```python
[38]: df['func_class'].value_counts()
      df['func_cat'] = df['func_class'].map({'Residential': 1,
                                             'Collector': 2,
                                             'Major': 3, 'Prime':4,
                                             'Local':5, 'Alley':6})
```

The AC Improved, PCC Jointed Concrete, AC Unimproved, and UnSurfaced pavement classes are converted to dummy variables.

```python
[39]: df['pvm_class'].value_counts()
      df['pvm_cat'] = df['pvm_class'].map({'AC Improved': 1,
                                           'PCC Jointed Concrete': 2,
                                           'AC Unimproved': 3,
                                           'UnSurfaced':4})
```

The current status of the job (i.e., post construction, design, bid/award, construction, and planning) is also converted to dummy variables.

```python
[40]: df['status'].value_counts()
      df['status_cat'] = df['status'].map({'post construction': 1,
                                           'design': 2,
                                           'bid / award': 3,
                                           'construction':4,
                                           'planning': 5})
```

17

Dropping Non-Useful/Re-classed Columns

Columns with explicit titles (i.e., names) and non-convertible/non-meaningful strings are dropped. Redundant columns (columns that have been cast to dummy variables) have also been dropped in conjunction with the index column which serves no purpose for this experiment.

```python
[41]: # drop unnecessary columns
df = df.drop(columns=['street_from',
                      'street_to',
                      'street_name',
                      'seg_id',
                      'street',
                      'pve_id',
                      'title',
                      'project_manager',
                      'project_manager_phone',
                      'project_id',
                      'resident_engineer',
                      'address_street',
                      'date_moratorium',
                      'OCI Range',
                      'total_count'])

df = df.reset_index(drop=True)

# drop variables exhibiting multicollinearity
df = df.drop(columns=['seg_length_ft',
                      'seg_width_ft',
                      'length',
                      'width',
                      'paving_miles',
                      'oci_wt'])

# drop re-classed columns
df = df.drop(columns=['func_class',
                      'pvm_class',
                      'status',
                      'type',
                      'date_end',
                      'date_start',
                      'oci_desc'])
```

The original dataframe is copied into a new dataframe *df1* in order to continue the final steps in the preprocessing endeavor. This is to avoid any mis-steps or adverse/unintended effects on the original dataframe.

```python
[42]: # create new dataframe for final pre-processing steps
df1 = df.copy()
```

One consequence of pre-processing data is that additional missing values may be brought into the mix, so one final sanity check for this phenomenom is commenced as follows.

```python
[43]: df_check = df.isna().sum()
df_check[df_check>0]
```

```
[43]: Series([], dtype: int64)
```

```
[44]: cor_matrix = df.corr().abs()
      upper_tri = cor_matrix.where(np.triu(np.ones(cor_matrix.shape),
                                            k=1).astype(np.bool))


      to_drop = [column for column in upper_tri.columns if
                 any(upper_tri[column] > 0.75)]


      print('These are the columns prescribed to be dropped: %s'%to_drop)
```

These are the columns prescribed to be dropped: []

Handling Class Imbalance

Multiple methods for balancing a dataset exist like "undersampling the majority classes" (Fregly & Barth, 2021, p. 178). To account for the large gap (95%) of mis-classed data on the "poor" condition class, "oversampling the minority class up to the majority class" (p. 179) is commenced. However, such endeavor cannot proceed in good faith without the unsupervised dimensionality reduction technique of Principal Component Analysis (PCA), which is carried out "to compact the dataset and eliminate irrelevant Features" (Naseriparsa & Kashani, 2014, p. 33). In this case, a new dataframe is reduced down into the first two principal components since the largest percent variance explained exists therein.

```
[45]: # the first two principal components are used
      pca = PCA(n_components=2, random_state=777)
      data_2d = pd.DataFrame(pca.fit_transform(df1.iloc[:,0:9]))
```

The dataframe is prepared for scatterplot analysis as follows.

```
[46]: data_2d = pd.concat([data_2d, df1['oci_cat']], axis=1)
      data_2d.columns = ['x', 'y', 'oci_cat']; data_2d
```

```
[46]:                 x       y  oci_cat
      0        7,986.11  -38.95        0
      1          47.96  -40.17        1
      2        -435.67  -39.63        1
      3         221.51  -39.71        1
      4        -348.92  -39.08        0
      ...           ...     ...      ...
      22993 -15,801.67  -40.53        1
      22994  12,768.33  114.26        1
      22995   9,128.33  114.06        1
      22996 -12,991.67  -40.48        1
      22997 -12,800.19  -40.25        1

      [22998 rows x 3 columns]
```

```
[47]: sns.lmplot('x','y', data_2d,
                  fit_reg=False,
                  hue='oci_cat',
                  palette=['#00BFC4',
                           '#F8766D'])
      plt.title('Class Imbalance in Street Condition Index'); plt.show()
```

Class Imbalance in Street Condition Index
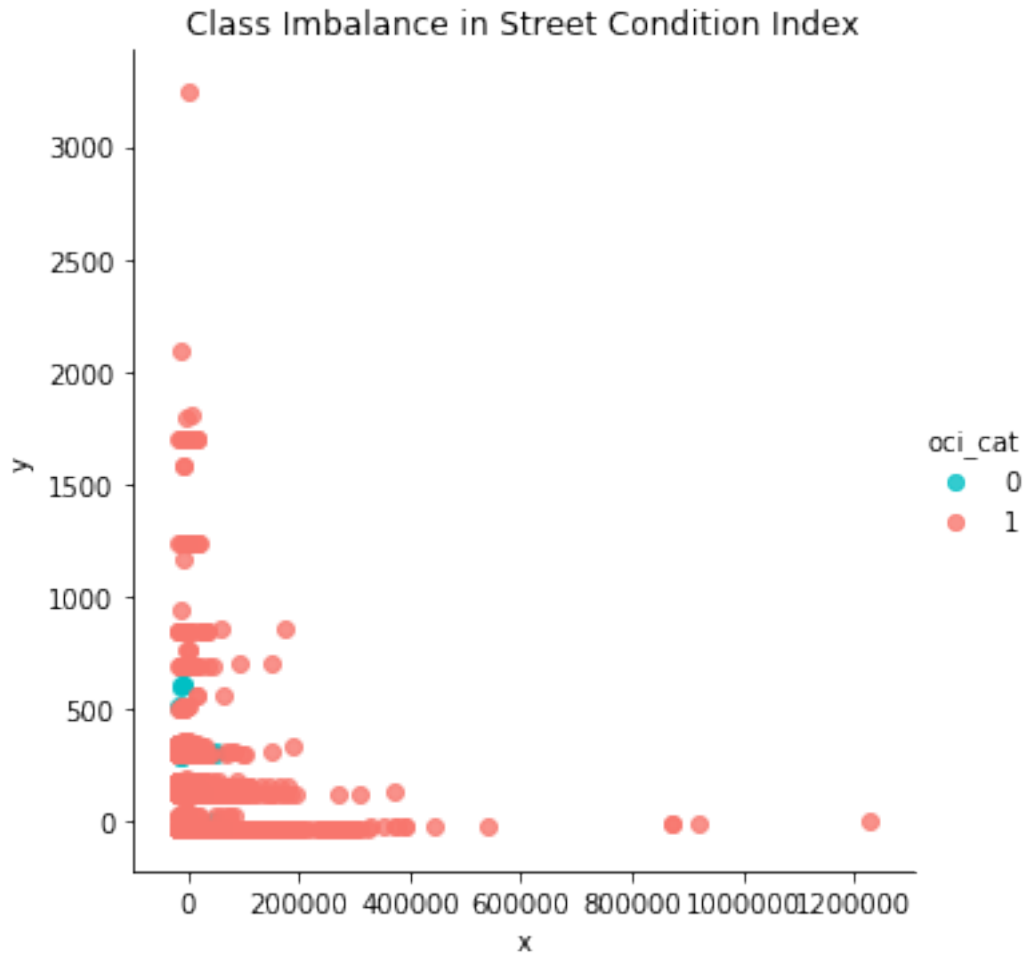
The dataset is oversampled into a new dataframe *df2*.

The adaptive synthetic sampling approach (ADAYSN) is leveraged "where more synthetic data is generated for minority class examples that are harder to learn compared to those minority examples that are easier to learn" (He et al., 2008). This allows for the minority class to be more closely matched (up-sampled) to the majority class for an approximately even 50/50 weight distribution.

```
[48]: ada = ADASYN(random_state=777)
      X_resampled, y_resampled = ada.fit_resample(df1.iloc[:,0:7],
                                                  df1['oci_cat'])
```

```
[49]: df2 = pd.concat([pd.DataFrame(X_resampled),
                       pd.DataFrame(y_resampled)], axis=1)
      df2.columns = df1.columns
```

The classes are re-balanced in a new dataframe using oversampling:

```
[50]: # rebalanced classes in new df
      df2['oci_cat'].value_counts()
      zero_count = df2['oci_cat'].value_counts()[0]
      one_count = df2['oci_cat'].value_counts()[1]
      zero_plus_one = zero_count + one_count
```

20

```
print('Poor Condition Size:', zero_count)
print('Good Condition Size:', one_count)
print('Total Condition Size:', zero_plus_one)
print('Percent in Poor Condition:', round(zero_count/zero_plus_one,2))
print('Percent in Good Condition:', round(one_count/zero_plus_one,2))
```

```
Poor Condition Size: 21714
Good Condition Size: 21858
Total Condition Size: 43572
Percent in Poor Condition: 0.5
Percent in Good Condition: 0.5
```

The dataframe can now be prepared as a flat .csv file if so desired.

Train-Test-Validation Split

[51]:
```
#Divide train set by .7, test set by .15, and valid set .15
size_train = 30500
size_valid = 6536
size_test = 6536
size_total = size_test + size_valid + size_train
train, test = train_test_split(df2, train_size = size_train,\
                               random_state = 777)
valid, test = train_test_split(test, train_size = size_valid,\
                               random_state = 777)

print('Training size:', size_train)
print('Validation size:', size_valid)
print('Test size:', size_test)
print('Total size:', size_train + size_valid + size_test)
print('Training percentage:', round(size_train/(size_total),2))
print('Validation percentage:', round(size_valid/(size_total),2))
print('Test percentage:', round(size_test/(size_total),2))
```

```
Training size: 30500
Validation size: 6536
Test size: 6536
Total size: 43572
Training percentage: 0.7
Validation percentage: 0.15
Test percentage: 0.15
```

[52]:
```
# define (list) the features
X_var = list(df2.columns)

# define the target
target ='oci_cat'
X_var.remove(target)
X_train = train[X_var]
y_train = train[target]
X_test = test[X_var]
y_test = test[target]
```

```
X_valid = valid[X_var]
y_valid = valid[target]
```

[53]:
```
# rearrange columns so that the target column is set up first
# for later training
df2 = df2[['oci_cat', 'oci', 'area_sq_ft', 'seg_cd', 'day_diff',
           'func_cat', 'pvm_cat', 'status_cat']]
```

[54]:
```
# reinspect the dataframe
df2.head()
```

[54]:

	oci_cat	oci	area_sq_ft	seg_cd	day_diff	func_cat	pvm_cat	status_cat
0	0	34.14	28,137.78	2	0	6	1	0
1	1	97.25	20,199.63	2	0	6	2	1
2	1	62.67	19,716.00	2	0	6	2	1
3	1	68.86	20,373.18	9	0	6	2	1
4	0	28.67	19,802.75	9	0	6	2	0

Transfer The Final Dataframe ($df2$) to S3 Bucket

[55]:
```
s3_client = boto3.client("s3")
BUCKET='waterteam1'
KEY='raw_files/df2/df2.csv'
response = s3_client.get_object(Bucket=BUCKET, Key=KEY)

with io.StringIO() as csv_buffer:
    df2.to_csv(csv_buffer, index=False, header=True)

    response = s3_client.put_object(
        Bucket=BUCKET, Key=KEY, Body=csv_buffer.getvalue()
    )
```

Modeling and Training

Logistic Regression

Herein, the classical Anaconda-based scikit-learn approach is leveraged to train the logistic regression model on the validation set.

[56]:
```
# Un-Tuned Logistic Regression Model
logit_reg = LogisticRegression(random_state=777)
logit_reg.fit(X_train, y_train)

# Predict on validation set
logit_reg_pred1 = logit_reg.predict(X_valid)

# accuracy and classification report (Untuned Model)
print('Untuned Logistic Regression Model')
print('Accuracy Score')
print(accuracy_score(y_valid, logit_reg_pred1))
print('Classification Report \n',
      classification_report(y_valid, logit_reg_pred1))
```

```
Untuned Logistic Regression Model
Accuracy Score
0.8959608323133414
Classification Report
              precision    recall  f1-score   support

           0       0.93      0.85      0.89      3286
           1       0.86      0.94      0.90      3250

    accuracy                           0.90      6536
   macro avg       0.90      0.90      0.90      6536
weighted avg       0.90      0.90      0.90      6536
```

Next, the logistic regression model is tuned using `RandomizedSearchCV()` and cross validated using repeated stratified kfold with five splits and two repeats. A set of hyperparamaters are subsequently defined to produce an overall best accuracy score in conjunction with a set of optimal hyperparameters.

```python
[57]: model1 = LogisticRegression(random_state=777)
      cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=2,
                                  random_state=777)
      space = dict()

      # define search space
      space['solver'] = ['newton-cg', 'lbfgs', 'liblinear']
      space['penalty'] = ['none', 'l1', 'l2', 'elasticnet']
      space['C'] = loguniform(1e-5, 100)

      # define search
      search = RandomizedSearchCV(model1, space,
                                  scoring='accuracy',
      n_jobs=-1, cv=cv, random_state=777)

      # execute search
      result = search.fit(X_train, y_train)

      # summarize result
      print('Best Score: %s' % result.best_score_)
      print('Best Hyperparameters: %s' % result.best_params_)
```

```
Best Score: 0.9518524590163935
Best Hyperparameters: {'C': 0.005639439254142048, 'penalty': 'l2', 'solver': 'lbfgs'}
```

Training, Testing, and Deploying a Model with Amazon SageMaker's Built-in XGBoost Model

```python
[58]: # Define IAM role
      role = get_execution_role()

      # set the region of the instance
      my_region = boto3.session.Session().region_name

      # this line automatically looks for the XGBoost image URI and
```

```python
# builds an XGBoost container.
xgboost_container = sagemaker.image_uris.retrieve("xgboost",
                                                  my_region,
                                                  "latest")


print("Success - the MySageMakerInstance is in the " + my_region + \
      " region. You will use the " + xgboost_container + \
      " container for your SageMaker endpoint.")
```

Success - the MySageMakerInstance is in the us-east-1 region. You will use the
811284229777.dkr.ecr.us-east-1.amazonaws.com/xgboost:latest container for your SageMaker
endpoint.

```python
[59]: train, test = np.split(df2.sample(frac=1, random_state=777),
                             [int(0.7 * len(df2))])


      print(train.shape, test.shape)
```

(30500, 8) (13072, 8)

Transfer The Training Data to S3 Bucket

```python
[60]: s3_client = boto3.client("s3")


      BUCKET='waterteam1'
      KEY='raw_files/train/train.csv'


      response = s3_client.get_object(Bucket=BUCKET, Key=KEY)


      with io.StringIO() as csv_buffer:
          train.to_csv(csv_buffer, index=False, header=False)

          response = s3_client.put_object(
              Bucket=BUCKET, Key=KEY, Body=csv_buffer.getvalue()
          )
```

```python
[61]: # input training parameters
      s3_input_train = sagemaker.inputs.TrainingInput(s3_data=\
              's3://{}/raw_files/train'.format(BUCKET), content_type='csv')
```

Setting up the SageMaker Session and Supplying Instance for XGBoost Model

```python
[62]: sess = sagemaker.Session()
      xgb = sagemaker.estimator.Estimator(xgboost_container,role,
                                          instance_count=1,
                                          instance_type='ml.m5.large',
                                          output_path='s3://{}/output'.format(BUCKET),
                                          sagemaker_session=sess)
      # parse in the hyperparameters
      xgb.set_hyperparameters(max_depth=5,eta=0.2,gamma=4,min_child_weight=6,
                              subsample=0.8,silent=0,
                              objective='binary:logistic',num_round=100)
```

Train The Model

[63]: `xgb.fit({'train': s3_input_train})`

2022-04-10 22:17:12 Starting - Starting the training job…
2022-04-10 22:17:29 Starting - Preparing the instances for
trainingProfilerReport-1649629032: InProgress
…
2022-04-10 22:18:55 Downloading - Downloading input data…
2022-04-10 22:19:56 Training - Downloading the training image..Arguments: train
[2022-04-10:22:20:25:INFO] Running standalone xgboost training.
[2022-04-10:22:20:25:INFO] Path /opt/ml/input/data/validation does not exist!
[2022-04-10:22:20:25:INFO] File size need to be processed in the node: 1.09mb.

Available memory size in the node: 294.12mb
[2022-04-10:22:20:25:INFO] Determined delimiter of CSV input is ','
[22:20:25] S3DistributionType set as FullyReplicated
[22:20:25] 30500x7 matrix with 213500 entries loaded from

/opt/ml/input/data/train?format=csv&label_column=0&delimiter=,
[22:20:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0

pruned nodes, max_depth=1
[0]#011train-error:3.3e-05
[22:20:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0

pruned nodes, max_depth=1
[1]#011train-error:3.3e-05
[22:20:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0

pruned nodes, max_depth=1
[2]#011train-error:0
[22:20:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0

pruned nodes, max_depth=1
[3]#011train-error:0
[22:20:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0

pruned nodes, max_depth=1
[4]#011train-error:0
[22:20:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0

pruned nodes, max_depth=1
[5]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0

pruned nodes, max_depth=1
[6]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0

pruned nodes, max_depth=1
[7]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0

pruned nodes, max_depth=1
[8]#011train-error:0

```
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0
pruned nodes, max_depth=1
[9]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0
pruned nodes, max_depth=1
[10]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0
pruned nodes, max_depth=1
[11]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0
pruned nodes, max_depth=1
[12]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0
pruned nodes, max_depth=1
[13]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0
pruned nodes, max_depth=1
[14]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0
pruned nodes, max_depth=1
[15]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0
pruned nodes, max_depth=1
[16]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0
pruned nodes, max_depth=1
[17]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0
pruned nodes, max_depth=1
[18]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0
pruned nodes, max_depth=1
[19]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0
pruned nodes, max_depth=1
[20]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0
pruned nodes, max_depth=1
[21]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0
pruned nodes, max_depth=1
[22]#011train-error:0
```

[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0 pruned nodes, max_depth=1
[23]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0 pruned nodes, max_depth=1
[24]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0 pruned nodes, max_depth=1
[25]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0 pruned nodes, max_depth=1
[26]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0 pruned nodes, max_depth=1
[27]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0 pruned nodes, max_depth=1
[28]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0 pruned nodes, max_depth=1
[29]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0 pruned nodes, max_depth=1
[30]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0 pruned nodes, max_depth=1
[31]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0 pruned nodes, max_depth=1
[32]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0 pruned nodes, max_depth=1
[33]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0 pruned nodes, max_depth=1
[34]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0 pruned nodes, max_depth=1
[35]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0 pruned nodes, max_depth=0
[36]#011train-error:0

```
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[37]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[38]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[39]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[40]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[41]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[42]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[43]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[44]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[45]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[46]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[47]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[48]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[49]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[50]#011train-error:0
```

```
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[51]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[52]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[53]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[54]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[55]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[56]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[57]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[58]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[59]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[60]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[61]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[62]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[63]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[64]#011train-error:0
```

[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0 pruned nodes, max_depth=0
[65]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0 pruned nodes, max_depth=0
[66]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0 pruned nodes, max_depth=0
[67]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0 pruned nodes, max_depth=0
[68]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0 pruned nodes, max_depth=0
[69]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0 pruned nodes, max_depth=0
[70]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0 pruned nodes, max_depth=0
[71]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0 pruned nodes, max_depth=0
[72]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0 pruned nodes, max_depth=0
[73]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0 pruned nodes, max_depth=0
[74]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0 pruned nodes, max_depth=0
[75]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0 pruned nodes, max_depth=0
[76]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0 pruned nodes, max_depth=0
[77]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0 pruned nodes, max_depth=0
[78]#011train-error:0

```
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[79]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[80]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[81]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[82]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[83]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[84]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[85]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[86]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[87]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[88]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[89]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[90]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[91]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[92]#011train-error:0
```

```
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[93]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[94]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[95]#011train-error:0
[22:20:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[96]#011train-error:0
[22:20:27] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[97]#011train-error:0
[22:20:27] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[98]#011train-error:0
[22:20:27] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0
pruned nodes, max_depth=0
[99]#011train-error:0

2022-04-10 22:20:43 Uploading - Uploading generated training model
2022-04-10 22:20:43 Completed - Training job completed
Training seconds: 118
Billable seconds: 118
```

Deploying The Predictor

```python
[64]: xgb_predictor = xgb.deploy(initial_instance_count=1,
                                 instance_type='ml.m5.large')
```

```
-----!
```

Running Predictions

```python
[65]: from sagemaker.serializers import CSVSerializer

      # load the data into an array
      test_array = test.drop(['oci_cat'], axis=1).values

      # set the serializer type
      xgb_predictor.serializer = CSVSerializer()

      # predict!
      predictions = xgb_predictor.predict(test_array).decode('utf-8')

      # and turn the prediction into an array
```

```
predictions_array = np.fromstring(predictions[1:], sep=',')
print(predictions_array.shape)
```

(13072,)

Evaluating The Model

```python
[66]: cm = pd.crosstab(index=test['oci_cat'],
                  columns=np.round(predictions_array),
                  rownames=['Observed'],
                  colnames=['Predicted'])
tn = cm.iloc[0,0]; fn = cm.iloc[1,0]; tp = cm.iloc[1,1];
fp = cm.iloc[0,1]; p = (tp+tn)/(tp+tn+fp+fn)*100
print("\n{0:<20}{1:<4.1f}%\n".format("Overall Classification Rate: ", p))
print("{0:<15}{1:<15}{2:>8}".format("Predicted", "Poor Condition",
                                    "Good Condition"))
print("Observed")
print("{0:<15}{1:<2.0f}% ({2:<}){3:>6.0f}% ({4:<})".format("Poor Condition", \
                                    tn/(tn+fn)*100,tn, fp/(tp+fp)*100, fp))
print("{0:<16}{1:<1.0f}% ({2:<}){3:>7.0f}% ({4:<}) \n".format("Good Condition", \
                                    fn/(tn+fn)*100,fn, tp/(tp+fp)*100, tp))
```

Overall Classification Rate: 100.0%

```
Predicted       Poor Condition Good Condition
Observed
Poor Condition 100% (6497)      0% (0)
Good Condition  0% (0)      100% (6575)
```

Terminating the Endpoint To Save on Costs

```python
[67]: # clean-up by deleteting endpoint
xgb_predictor.delete_endpoint(delete_endpoint_config=True)
```

References

Amazon Web Services. (n.d.). *Amazon Athena.*
https://aws.amazon.com/athena/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc

Amazon Web Services. (n.d.). *Build, train, and deploy a machine learning model with Amazon SageMaker.*
https://aws.amazon.com/getting-started/hands-on/build-train-deploy-machine-learning-model-sagemaker/

Fregly, C., & Barth, A. (2021). *Data Science on AWS.* O'Reilly.

Garrick, D. (2021, September 12). San Diego to spend $700K assessing street conditions to spend repair money wisely. *The San Diego Union-Tribune.*
https://www.sandiegouniontribune.com/news/politics/story/2021-09-12/san-diego-to-spend-700k-assessing-street-conditions-to-spend-repair-money-wisely

He, H., Bai, Y., Garcia, E. & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning.
2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational

Intelligence),1322-1328.
https://ieeexplore.ieee.org/document/4633969

Naseriparsa, M. & Kashani, M.M.R. (2014). Combination of PCA with SMOTE Resampling to Boost the Prediction Rate in Lung Cancer Dataset.
International Journal of Computer Applications, 77(3) 33-38. https://doi.org/10.5120/13376-0987