“Call Type” Predictions for Fire Department Calls for the city of San Francisco

Department of Information Systems, California State University Los Angeles
David Ward
Shivaniben Shah
Tanjina Akter Reema
Yangyang Jia

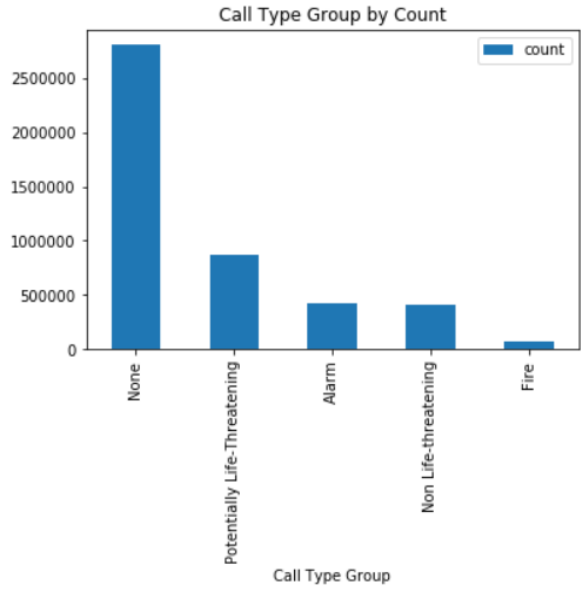
Abstract:

The project focuses to using historical records from Year 2000 to Year 2018 of Fire Department Calls of city of San Francisco to predict(classify) “Call Type” for a specific location for certain day and time of week. The main goal of the project is to predict(classify) “Call Type” from different types such as ‘Explosion’, ‘Fuel Spill’, ‘Gas Leak’, ‘Outside Fire’, ‘Water Rescue’, ‘Traffic Collision’ etc for a call to Fire Department of San Francisco. The use-case will provide more insights to respective fire departments to take important decisions to improve their services and save more lives by predicting “Call Type” for future calls.

Introduction

Fire Calls-For-Service includes all fire units responses to calls. Each record includes the call number, incident number, address, unit identifier, call type, and disposition. Dataset Size has Rows: 4.61 Million, Columns: 34, Data: From Year 2000 to 2018 and Size in GBs: 1.9 GB. All relevant time intervals are also included. Because this dataset is based on responses, and since most calls involved multiple units, there are multiple records for each call number. Addresses are associated with a block number, intersection or call box, not a specific address.

Call Type Group
Alarm
null
Potentially Life-...
Non Life-threatening
Fire

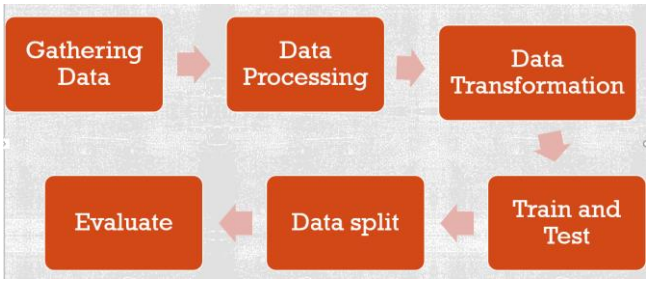


In Data Pre-processing step, null values – missing values and outliers that can adversely affect the predicted(classified) value will be taken care of. Data Sampling techniques may also be applied for large datasets. In Data Exploration step, various trends for various categories of “Call Type Group” can be visualized (through various statistical measures) to understand importance of available features and hidden relations among them. May be these relations and trends can be discussed with domain experts to understand problem at hand at deep level. In Feature Extraction and Engineering step, from available features one can extract new features using dimensionality reduction techniques – PCA, SVD, etc OR feature engineering techniques - Date and time features, numeric to categorical mappings, grouping sparse classes, etc. that are more suitable for prediction task to achieve maximum accuracy. In Model(s) Training and Testing step, selected appropriate models for classification tasks – SVM, Decision Trees, KNN, Neural Network, etc. can be trained using transformed features obtained through feature engineering step.

Dataset will be divided in Training and Testing datasets, then again Training data is divided in Training and Validation data. Model(s) In Evaluation and Selection, based on performance of model(s) on Training, validation and testing dataset (unseen data), it will be evaluated and model that provides highest accuracy measure (f-score, roc curve, precision, recall and accuracy) in cross validation and testing dataset is selected for classification purpose. In Knowledge mining and Visualizations step, based on predicted(classified) category of "Call Type Group" for a certain location on a certain day and time – Fire department can take certain actions and execute certain decisions to enhance their services and save more lives.

Methodology

Here are the simple flow of implementing the project. That is loading dataset into the Linux Environment and analyzing the structure of the dataset.



- We build our model with Spark ML in Databricks2 . Here, we run our model with the best algorithms from the first part and try to improve our result taking the whole dataset into considerationFirst, we gather all the data and did the data cleaning process so we did that in Databricks. Also we add the file into the databricks which provides a Unified Analytics Platform that accelerates innovation by integrating data science, engineering and business . So it provides an ease to manage big data in more simple and efficient way.
- According to our use case, found out irrelevant features and removed them, such as - 'call_number', 'unit_id', 'rowid' etc.Attribute such as - 'city' contains values with different representation for same values such as 'San Francisco', 'SAN FRANCISCO', 'SF' for city of San Francisco. So we have retained only single representation for those values.
- Attributes such as - 'Supervisor_district', 'Box' and 'Station_area' contains string representation as well as numeric representation for same values. So we have retained only string representation for 'supervisor_district', 'box' and 'station_area.
- Understood the context of features and removed features which can only be known after providing response to a call and do not contribute to predict 'call_type' - such as 'dispatch_dttm', 'entry_dttm', 'hospital_dttm', 'on_scene_dttm', 'response_dttm', 'transport_dttm', 'available_dttm', 'call_final_disposition', 'final_priority', 'priority', 'als_unit', 'unit_type', 'unit_sequence_in_call_dispatch'.
- Understood the context of features and removed repetitive features (duplicate features) such as - 'call_date', 'watch_date' which can be combinely represented using 'received_dttm'.
- Considered following 14 attributes for further exploration: - 'Incident_number' - 'call_type_group' - 'received_dttm' - 'address' - 'zipcode_of_incident' - 'battalion' - 'station_area' - 'box' - 'original_priority' 8 - 'number_of_alarms' - 'location' - 'fire_prevention_district' - 'supervisor_district' - 'neighborhoods_analysis_boundaries'.
- Removed the duplicate instances involved in refined dataset after removing the irrelevant features. Also, eliminated the missing values, null values for remaining 14 attributes. After removing those samples, still we have nearly 2 Millions of samples to train model.

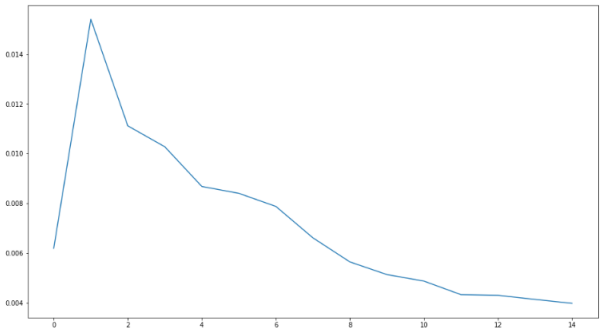
  From available features we extract new features using dimensionality reduction techniques - Date and time features, numeric to categorical mappings, grouping sparse classes, etc. that are more suitable for prediction task to achieve maximum accuracy. These columns are selected to avoid data leakage in our modeling. Data leakage is common in

modeling and can lead to invalid predictive models .

```
from pyspark.sql import functions as F
fireIndicator = df.select(df["Call Type"],F.when(df["Call Type"].like("%Fire%"),1)\
                          .otherwise(0).alias('Fire Indicator'))
fireIndicator.show()
```

```
+--------------------+--------------+
|          Call Type|Fire Indicator|
+--------------------+--------------+
|    Medical Incident|             0|
|    Medical Incident|             0|
|    Medical Incident|             0|
|              Alarms|             0|
|    Medical Incident|             0|
|Citizen Assist / ...|             0|
|   Electrical Hazard|             0|
|Odor (Strange / U...|             0|
|    Medical Incident|             0|
|    Medical Incident|             0|
|              Alarms|             0|
|    Medical Incident|             0|
|    Medical Incident|             0|
|    Medical Incident|             0|
|              Alarms|             0|
|              Alarms|             0|
|    Medical Incident|             0|
|               Other|             0|
|    Medical Incident|             0|
|     Structure Fire|              1|
+--------------------+--------------+
only showing top 20 rows
```

- Later, Identified that 'Box' and 'Supervisor District' includes float, int and string data types making it hard for using as feature. Converted all instances of 'Box' and 'Supervisor District' to string.
- Converted 'Received DtTm' feature into weekday, hour and month using user defined function. Defined three functions namely convert_ date_to_weekday, convert_date_to_hour, convert_date_to_month.
- Utilized 'Station Area' feature initially, but due to sparse distribution and less variable importance of 'Station Area', we finally removed the feature.
- Tried to apply "One hot encoding" for "Address" attribute. Nearly 27000 unique values of "Address" attribute caused "Memory Error" while creating dense data frame while applying "One hot encoding". So, used Compressed Sparse Row Matrix data structure to efficiently utilize the memory.
- Using "Address", "Box", "Month", "Weekday", "Hour", and " Neighborhoods_ analysis_boundaries" created a CSR sparse matrix that can efficiently utilize the memory.
- Applied dimensionality reduction technique namely "Truncated SVD" on CSR matrix. Plotted explained variance versus number of components graph to figure out inflection point and to choose n_components parameter.



The dataset has to be normalized and split into a train and test set, respectively 70% and 30%.For Model Training and Testing, we divided the dataset into Training Dataset - 1.80 M and Testing dataset - 0.26 M. Trained and tested following models: Logistic Regression, Decision Tree and SVM(Support Vector Mechanism)

Model Name	Training Accuracy (F-1 Score)	Testing Accuracy (F-1 Score)
MLP Classifier	0.71	0.759
Random Forest Classifier	0.71	0.76
XGBoost Classifier	0.71	0.759
SVM Classifier	0.71	0.759

There are three main metrics which are important in terms of evaluating the model. The accuracy, precision and recall. In our case, the recall is the most important metric because the aim of this classification problem is to detect fraudulent transactions. A good recall implies that the model is good in predicting whether we got a call for fire or not.

Model	Accuracy	Precision	Recall
Logistic Regression	0.938	0.912	0.963
Decision Tree	0.566	0.765	0.867
SVM	0.777	0.789	0.877

Clearly, the Logistic Regression is the best algorithm for our model since it has the highest recall score also with good accuracy. Based on this results we will continue building our model in Databricks mainly with the DF.

We took the whole dataset and tried three different classification models. Classification models work best when all the features are combined in a single vector for training purposes. Therefore, we begin the vectorization process by collecting all of the features into a single list called features. In order to minimize overfitting the model, the dataframe will be split into a testing and training dataset to fit the model on the training dataset, trainDF and test on the testing dataset, testDF. We used the Random Forest Classifier (RF), the Decision Tree Classifier (DT) and Logistic Regression (LR).

```
+-----+--------------------+--------------------+--------------------+----------+
|label|            features|       rawPrediction|         probability|prediction|
+-----+--------------------+--------------------+--------------------+----------+
|    0|(9,[0,1,2,3],[1.0...|[2.60599500917470...|[0.93124641306604...|       0.0|
|    0|(9,[0,1,2,3],[1.0...|[2.60599500917470...|[0.93124641306604...|       0.0|
|    0|(9,[0,1,2,3],[1.0...|[2.60599500917470...|[0.93124641306604...|       0.0|
|    0|(9,[0,1,2,3],[1.0...|[2.57706663898438...|[0.92937096394875...|       0.0|
|    0|(9,[0,1,2,3],[1.0...|[2.57706663898438...|[0.92937096394875...|       0.0|
+-----+--------------------+--------------------+--------------------+----------+
only showing top 5 rows
```

We calculated the accuracy of all the three model where logistic regression tend indicated good accuracy.

```
metrics.accuracy_score(actual, predicted)
```

```
0.88427350472115218
```

- Achieved 88% accuracy for Testing dataset in predicting the ‘call type’ for incoming calls for San Francisco Fire department.
- Learned the importance of Data Preprocessing and Feature engineering (Feature Extraction, Feature Transformation, Feature Reduction) in case of real-life dataset.
- The prediction of “Call Type” may help respective Fire Department of city of San Francisco to predict the type of situation beforehand at the place of incident. So, they can allocate resources Optimizely and can manage more than one incident in their area efficiently, to save precious lives of people.

References

[1]https://github.com/shivanishah03/Fire_Department

[2]https://gallery.cortanaintelligence.com/Experiment/Fire-department

[3]https://www.statisticssolutions.com/assumptions-of-logistic-regression/

[4]https://www.statisticssolutions.com/assumptions-of-multiple-linear-regression/

[5]https://data.sfgov.org/Public-Safety/Fire-Department-Calls-for-Service/nuek-vuh3

[6]https://www.statisticssolutions.com/assumptions-of-multiple-linear-regression/

[7]http://gradientdissent.com/blog/analyzing-2-months-of-real-crime-data-from-san-francisco-and-seattle.html#.W_3u1vZFxPY

[8]https://planspace.org/20150423forward_selection_with_statsmodels/

[9]https://github.com/asherif844/ApacheSparkDeepLearningCookbook

[10]https://github.com/vishwesh07/Fire-department-call-prediction/blob/master/CMPE_256_Project_Proposal.pdf