ENHANCING FIRE INCIDENT RESPONSE IN LONDON: UTILIZING DATA MINING TO PREDICT PEOPLE'S SAFETY IMPACT AND IMPROVE SAFETY MEASURES FOR WHITE GOODS-RELATED FIRES

ABSTRACT

This journal analyses the London Fire Brigade (LFB) fires in London that are caused by white goods like washing machines, dishwashers, refrigerators, etc and the impact on people through injuries and deaths.

The project uses a dataset from the London Datastore to predict death/injuries from fire incidents caused by white goods while considering the number of deaths caused by fire, number of injuries caused by fire and borough, ignition source, and location where the fire started as factors. It uses the CRISP-DM methodology to clean and obtain meaningful information from the dataset. The obtained data is now used for building models using machine learning models such as Random Forest Classifiers and K-Means clustering.

Many problems have risen from deaths and injuries to people, as it causes pain to their well-wishers and loved ones. Therefore, these problems keep begging for answers. Some of these questions include: How does fire caused by white goods contribute to deaths and injuries? Which white goods cause more fire? Which appliance manufacturer has their product caused more fire?

Random Forest Classifier and K-Means algorithms have shown to be very suitable in predicting if there will be injuries or deaths with their very high accuracy and silhouette score respectively.

Keywords: K-Means Clustering, CRISP-DM, Random Forest Clustering, LFB.

1.0 INTRODUCTION

1.1 LITERATURE REVIEW

The issue of deaths and injuries caused by fire incidents has always been a big issue. Apart from the dent it leaves on family and loved ones, it also cost the government in terms of money. Also, the time spent by medical practitioners in treating victims of fire occurrences could have been used to treat other cases in the hospital. This project tends to look at variables of factors like types of white goods, their manufacturers, and location among other factors to predict if people are going to be affected in a fire incident in terms of injuries and deaths. Not much research has gone into giving context or solutions to the death/injuries of people. While researching previous works, there was next to nothing on predicting injuries or deaths resulting from a fire caused by white goods. This project intends to close the gap in the lack of research works on the effect of fire incidents caused by white goods and how it affects Humans.

The Fire Brigades Union, a professional voice of firefighters and other workers within fire and rescue services across the UK, “every second counts”. The effort needed to put out a fire may worsen quickly Fire spreading quickly can in turn cause more deaths and the destruction of properties [1].

A study by Asor et al. [2] worked on four different machine learning algorithms namely random forest, decision tree, support vector, and recurrent neural network (RNN) to develop a model for pattern recognition and visualization of fire incidents in the province of Laguna, Philippines. The recurrent neural network was preferred because it was noticed to possess a better and more precise model as it provided a quality pattern in predicting fire incidents in the province. The author argued that the RNN model can be used in developing a machine learning model (ML) that can learn patterns related to fire.

R. Rishickesh, et al. [3] on the other hand predicted forest fire using classification techniques like Logistic Regression, Support Vector Machine and Random Forest, and ensemble Bagging and Boosting classifiers. It was discovered that with Principal Component Analysis (PCA), Logistic regression was the best among the models considered because it had the highest rate of accuracy.

Jaafari et al. [4] analyzed the spatial pattern of wildfires in the Zagros Mountains in Iran with five decision tree-based classifiers which are the function tree (FT), Naïve Bayes tree (NBT), alternating decision tree (ADT), classification and regression tree (CART) and Logistic model tree (LMT). The classifiers were validated using many statistical index-based evaluators which include sensitivity, accuracy, specificity, precision, and F-measure. The performance of the models was measured with the ROC-AUC method with the ADT classifier having the best fit.

B. T. Pham et al. [5] in their study to predict and map fire susceptibility across the Pu Mat National Park in Vietnam considered Bayes Network (BN), Decision Tree (DT), and Multivariate Logistic Regression (MLP) machine learning methods. There were differences between the AUC values, the Bayes Network model had the highest AUC value and was considered dominant over other models in predicting future fires. Closely followed was the Decision Tree, then the Naïve Bayes, and lastly the Multivariate Logistic Regression.

Choi et al. [6] in their study applied statistical machine learning and optimized risk indexing models for fire risk assessment. The fire risk indexing method is known as KFPA Fire Risk Index (KFRI). The statistical machine learning models are Logistic Regression and Deep Neural Networks. DNN and Logistic Regression were first used for fire occurrence prediction. The prediction accuracy of the DNN was found to be better than KFRI and Logistic Regression.

1.2 SUMMARY OF DATA MINING RESULTS

From the results, it can be deduced that some boroughs have very few fire incidents but more injuries and deaths. Case in point, Kensington and Chelsea had 18.8% of fire injuries for all the fires that were caused by white goods between 2009 and 2022 and a whopping 85.5% of deaths in London within that period. But one thing that should be accounted for is the unfortunate Grenfell Tower incident that happened in 2017. This incident accounted for most of the deaths and injuries. Even at such, Kensington and Chelsea had some of the lowest fire incidents caused by white goods among the boroughs in London.

Brent, Bromley, together with Kensington and Chelsea accounted for a whopping 95.1% of all fire casualties during that period. Kensington and Chelsea, Croydon, Lewisham, Bromley, and Ealing accounted for 41.9% of injuries during that period. Croydon and Southwark accounted for most fire incidents and surprisingly do not feature in the top 3 of deaths and injuries.

When it comes to manufacturers, Bosch, Beko, Indesit, and Hotpoint account for 41.6 of all white goods that cause fire incidents. This points to the possibility that their goods may be full of defaults. The white goods that cause the most incidents of fire were discovered to be the washing machine with more than 1400 incidents and then the Tumble Dryer-Standard. All these put together can help decision-makers pinpoint the type of white goods causing fire incidents and their manufacturers.

2.0 METHODOLOGY

2.1 DATA UNDERSTANDING

This project follows the CRISP-DM methodology. What the project hopes to achieve has been discussed in the abstract section of this journal. The next step is to understand the data better.

Source of the data: Fires in London - cause of ignition is white goods https://data.london.gov.uk/dataset/fires-in-london---cause-of-ignition-is-white-goods.

Size: The dataset contains 4,438 data points and 17 attributes of data.

Link of Colab: https://colab.research.google.com/drive/1fGpyzWki4HhzSPhKS3IkZi7qi4laHOak?usp=sharing.

Type of Data: Comma-separated values (CSV) file format with alphanumeric fields.

Year of Collection: 2023

Information Contained: The dataset contains details of fires that are caused by the ignition of white goods (e.g., washing machine, dishwasher, refrigerator).

Variables in Dataset:

Year: The year of the fire incident.

Month: The month of the fire incident.

IncType: The type of fire incident.

ParentPropertyType: The type of property where the fire incident occurred.

NumFireDeaths: The number of deaths from the fire incident.

NumAllFireInjuries: The total number of injuries resulting from the fire incident.

IncGeo_BoroughCode: The code for the borough where the fire incident occurred.

IncGeo_BoroughName: The name of the borough where the fire incident occurred.

IncGeo_WardCode: The code for the ward where the fire incident occurred.

IncGeo_WardName: The name of the ward where the fire incident occurred.

IgnitionSourcePower: The power source of the ignition that caused the fire incident.

IgnitionSource: The source of the ignition that caused the fire incident.

ItemFirstIgnited: The item or material that first caught fire in the incident.

LocationFireStarted: The specific location where the fire started.

ApplianceManufacturer: The manufacturer of the appliance that was involved in the fire incident, if applicable.

ApplianceManufacturerOther: Any other information about the appliance manufacturer, if applicable.

MainCauseModel: The main cause of the fire incident.

2.2 DATA PREPARATION

Data preparation involves data cleaning or data pre-processing and is very important. The data is being transformed into a format that is suitable for analysis and modeling. The first thing was to find the missing columns. The next step was creating a new column called 'NumPeopleAffected' from the combination of 'NumFireDeaths' and 'NumAllFireInjuries'.It checks if there is any death or injury and if there is at least 1, the new column is assigned 1 and if there is none, it is assigned 0. This new column 'NumPeopleAffected' is going to be the target variable.

The next step is removing uncorrelated columns and columns with a lot of empty entries as they may introduce bias into the results. This can be done using a Correlation matrix for numerical variables and a Chi-square test for categorical data.

After running the correlation matrix and chi-squared, some variables will be dropped. These are Year, IncType, IgnitionSourcePower, Month, ApplianceManufacturerOther, IncGeo_WardName, IncGeo_WardCode, IncGeo_BoroughCode, and MainCauseModel.

The remaining columns are ParentPropertyType, NumFireDeaths, NumAllFireInjuries, NumPeopleAffected, IncGeo_BoroughName, IgnitionSource, ItemFirstIgnited, LocationFireStarted, and ApplianceManufacturer.

In the next step of fire cleaning, there is going to be a loop through all columns to check for irregular and missing data. The ApplianceManufacturer column had 485 null columns. Since there are 4,438 data points in the dataset and 485 represent more than 10% of the dataset, it won't be advisable to delete them to avoid loss of data and bias. The best thing was to change all those 485 null values and give them their category named 'Other'. Bear in mind that there is an 'Other' column already in the dataset. There were also a lot of values with 'Null' and they were also changed and grouped with 'Other'.

Another issue was in the 'IncGeo_BoroughName' column which had an issue with the case. Some were all upper-case, and the others were Camel Cases. In this scenario, all the upper-case occurrences were converted to Camel Case.

The final step was to one-hot encode or label encode all the categorical variables in the dataset into numerical variables. This was to help with our Machine Learning models predictions.

2.2.1 Data Exploration

Data Exploration involves checking the dataset both before and after cleaning for patterns in the dataset. These patterns tend to help decision-makers to be able to effectively reduce the loss of life. There will be plots of pie charts and histograms.

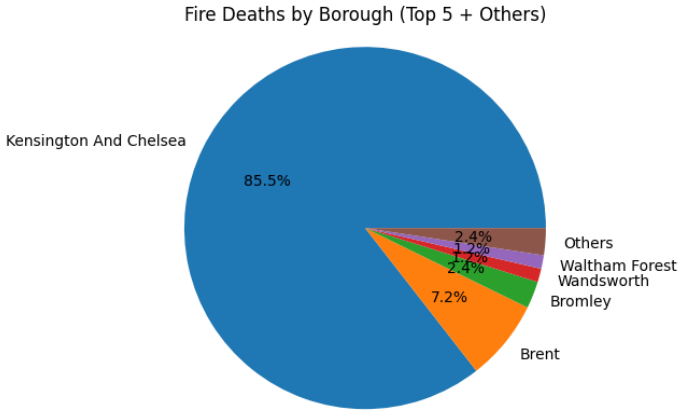

Figure 1: Pie Chart of deaths by borough

A pie chart showing the distribution of the deaths by borough. In the above, Bromley had the highest number of deaths resulting from fire incidents caused by the ignition of white goods within the time frame. Bromley, Merton, Harrow, and Waltham Forest account for all other deaths accounted for.

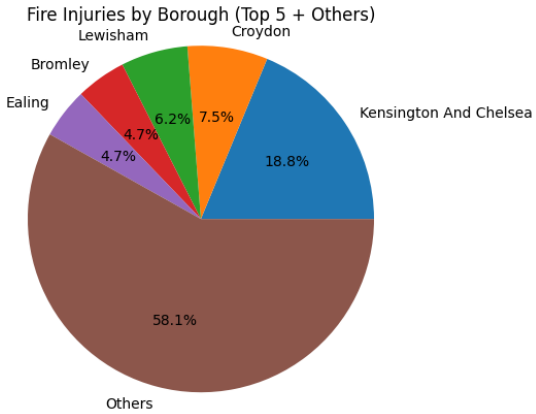

Figure 2: Pie Chart of injuries by Borough

The above is a pie of people that were injured from fire incidents caused by white goods. The most injuries were from Kensington and Chelsea,

Lewisham, and Bromley. With Bromley and Ealing completing the top 5.

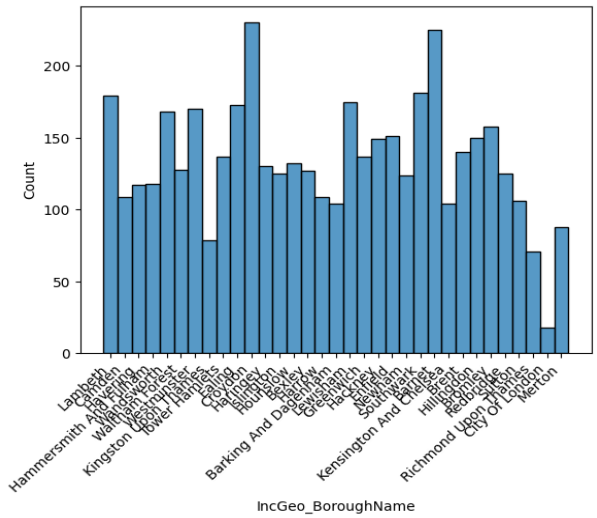

Figure 3: Borough by number of occurrences

The histogram shows the number of occurrences of fire incidents caused by white goods. With the most in Croydon and Brent.

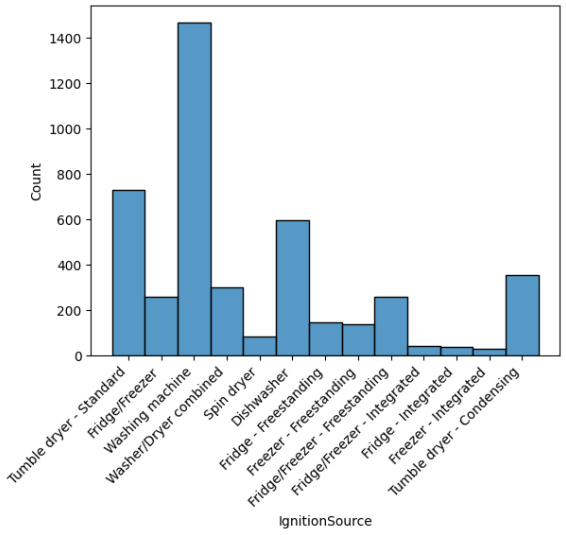

Figure 4: Fire incidents by Ignition Source

The histogram shows fire incidents by their ignition source. The washing machine shows to be the most capable of starting a fire followed by a Tumble Dryer-Standard and then a Dishwasher.

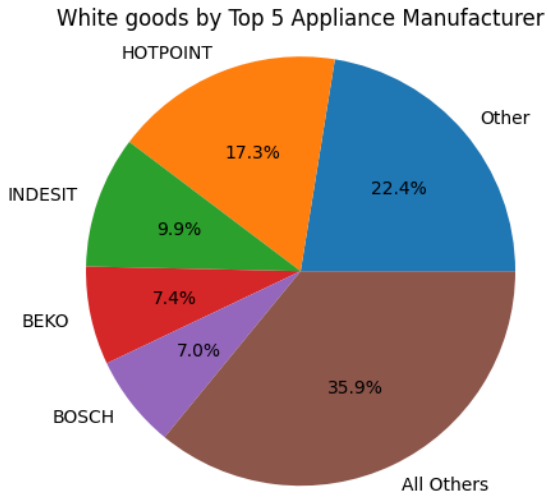

Figure 5: White goods by Top 5 Appliance Manufacturers

The pie chart shows the top 5 manufacturers of white goods that cause fire incidents. Hotpoint, Indesit, Beko, and Bosch tend to be the manufacturers of fire-causing white goods.

2.3 SUPERVISED LEARNING MODEL

The supervised learning model used for the analysis is the Random Forest Classifier which is a combination of multiple decision trees to make predictions. It builds decision trees on different subsets of the training data. Each decision tree is trained to predict the target variable. These decision trees combine to make a final prediction. Random forest classifiers have a very high level of accuracy and are also relatively easy to train and interpret. Random forest classifiers are often used for classification problems.

Random Forest Classifier was chosen because of its accuracy. They can attain very high accuracy even when the training data is with some certain level of bias. Another reason why it was chosen is because of its robustness as they are less likely to overfit the training data than other machine learning models.

The Python Libraries used to fit the model and evaluation are

Pandas: Pandas is a Python library that is used for data manipulation and analysis provides DataFrames and is used for pre-processing.

Scikit-Learn (Sklearn): A library used in Python that provides the Random Forest Algorithm and also houses the accuracy, precision, recall, F1-score, ROC curve, classification report, and confusion matrix.

Matplotlib: Matplotlib is used for data and results visualization, and it can be used to plot ROC Curves and evaluate the algorithm's performance.

Seaborn: It is a library for visualization, and it can be used to plot heatmaps such as confusion matrix.

Imbalanced-Learn: This is a library used to perform under-sampling of imbalanced datasets.

The software used for the evaluation was Google's Colab which is used to run Machine Learning models.

2.3.1 DESCRIPTION OF THE PARAMETERS USED TO FIT THE MODEL

random_state: This parameter is optional. This helps to ensure that when running the model multiple times, the model can obtain consistent results.

2.4 UNSUPERVISED LEARNING MODEL

The Unsupervised learning model used for the analysis is K-means Clustering. It is a machine learning algorithm that helps to group data points into clusters. There is a mean and the data points are assigned to the cluster with the centroid. This continues until there is no change in the centroid of the cluster. K-means Cluster is known for its simplicity as it is a relatively simple algorithm to understand and implement. They are also very flexible as they can cluster data points of any type. The objective of the WCSS is to minimize the within-cluster-sum of squares. The lower the WCSS, the better.

K-means Clustering was chosen because of its simplicity and flexibility. They are easy to understand and can cluster data points of any type.

The Python Libraries used to fit the model and evaluation are

Pandas: Pandas is a Python library that is used for data manipulation and analysis provides DataFrames and is used for pre-processing.

Scikit-Learn (Sklearn): A library used in Python that provides clustering algorithms like K-Means.

Matplotlib: Matplotlib is used for data and results visualization, and it can be used to visualize clusters and evaluate the algorithm's performance. It also contains the 3D plot used to plot the clusters' plot

The software used for the evaluation was Google's Colab which is used to run Machine Learning models.

2.4.1 DESCRIPTION OF THE PARAMETERS USED TO FIT THE MODEL

n_clusters: K is the number of clusters that the model will divide the data points into. This parameter determines the number of clusters that the model created. In this case, the n_clusters was set to 2 which means that the model should group the data into two clusters.

random_state: This parameter is optional. This helps to ensure that when running the model multiple times, the model can obtain consistent results.

2.4.2 HOW THE UNSUPERVISED METHOD (K-MEANS) HELPS IN THE IMPLEMENTATION OF THE SUPERVISED LEARNING (RANDOM FOREST)

One way K-means clustering helps to implement Random Forest is through pre-processing. This involves grouping similar data points which then helps in identifying patterns and structures in the dataset. Another way is in the area of feature engineering to derive new variables that can then be used for the Random Forest Classifier. These new variables can in turn be used as categorical variables. Another was in the area of data visualization to visualize how data is distributed and identify potential clusters.

3.0 CODE

See appendix

4.0 RESULTS AND DISCUSSION

4.1 UNSUPERVISED LEARNING METHOD

To implement the K-Means model, we first label and encode the categorical data into numerical variables. Then there is an implementation of the code. The clustering algorithm uses the elbow method to pick the best number for the value of K or n_clusters. The elbow method plots the within-cluster sum of squares (WCSS) against the number of clusters. The best was discovered to be 10 as the lower the WCSS, the better.

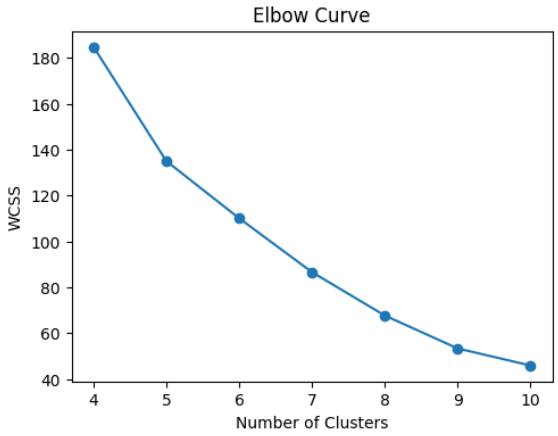

Figure 6: The Elbow Method

From the implementation, the Silhouette Score is 0.7078 or 70.78%. The Silhouette Score measures how well-defined the clusters are in the algorithm. The higher the score, the better the alignment of the clusters. A score of 0.7078 implies that the algorithm has reasonable clusters and that the clusters are well separated from each other.

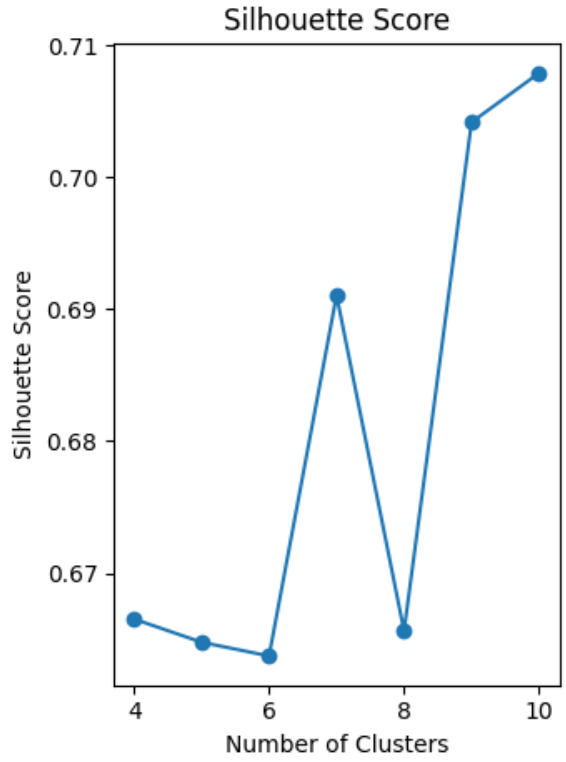

Figure 7: The Silhouette score

The IgnitionSource, LocationFireStarted, and NumPeopleAffected were all plotted against each other on a cluster plot. The cluster center was set, and the visualization is shown below.

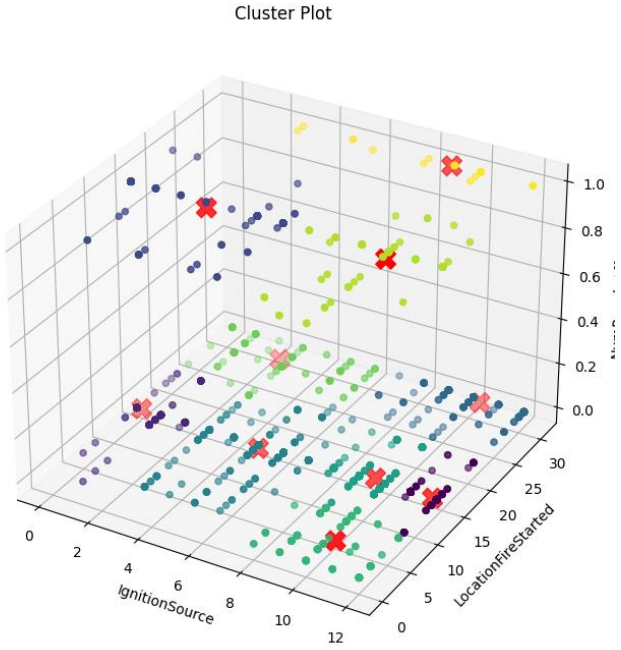

Figure 8: The Cluster Plot

From the above, it can be seen that 10 clusters are well packed together thanks to a low WCSS.

4.2 SUPERVISED LEARNING METHOD

The Random Forest Classifier was undersampled because there was a huge bias in the class as there were way more class 0s than class 1 in the target variable. The model can correctly classify in 99.18% of the data instances. The precision score is 98% which indicates that the model can avoid false positives 98% of the time. The recall is 100% which means at every instance there are no false negatives. The harmonic mean of 0.9898 which is also the F1-Score of 0.9898 means the model is performing well. Generally, the model performs very well and can accurately classify the data instances. With the performance of the model, it is recommended to evaluate the performance of the classifier on multiple datasets to ensure that the model is reliable.

4.3 MODEL ASSESSMENT OF SUPERVISED LEARNING

For the supervised Learning model Random Forest Classifier. The model assessment for supervised learning used is the accuracy, precision, recall F-score, ROC curve, and Classification report.

The accuracy which deals with the overall correctness of the classifier is 99.18% or 0.9918 which means out of 100 times, the Classifier correctly classifies instances.

The recall deals with correctly predicting positive values at a value of 1.0. It means it correctly predicts

all positive instances 100% of the time and there is no false negative.

The precision deals with the proportion of positive instances that were correctly predicted out of the total instances predicted as positive. It has a value of 98% which indicates that out of all the instances predicted as positive, there was a precision of 98%.

The F1-Score is the harmonic mean of the precision and recall. It considers both False positives and false negatives and has a value of 0.9898 or 98.98% which means the Random Forest Classifier performs well.

	Precision	Recall	f1-score	support
0	1.00	0.99	0.99	73
1	0.98	1.00	0.99	48

Table 1: Classification Report

The classification report shows the precision, recall, and f1-score for class 1 and class 0. The precision is 1 for class 0 and 0.98 for class 1. The recall is 0.99 for class 0 and 1.00 for class 1. The f score is 0.99.

The overall performance of the model is very good as it has an accuracy of 99.18% which means it correctly predicts a correct instance in 99.18% of its prediction at any time.

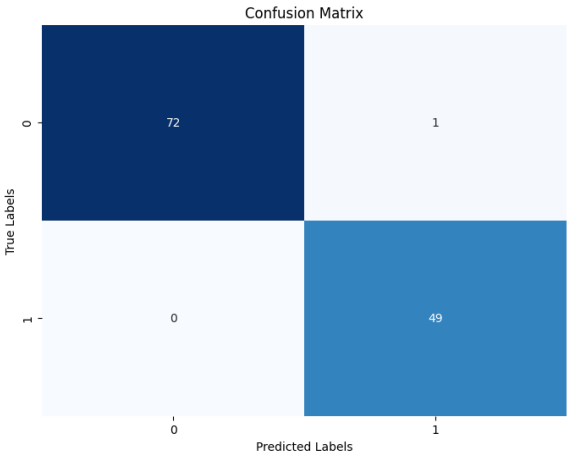

Figure 9: Confusion Matrix of the Random Forest Classifier

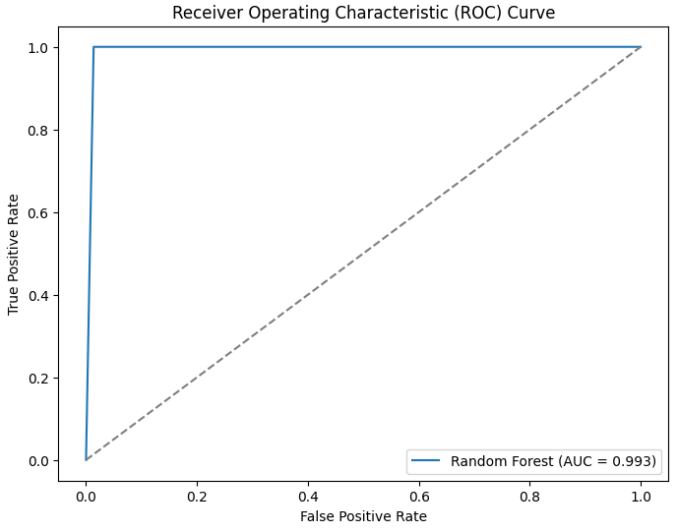

Figure 10: The Receiver Operating Characteristic Curve

4.4 MODEL ASSESSMENT OF UNSUPERVISED LEARNING MODEL

The K-means algorithm has a WCSS score of 40, a k of 10 clusters, and a Silhouette of 70.78% or 0.7078. The model performs very well as it has a WCSS of 40 which is the compactness and tightness of the clusters. A silhouette score of 0.7078 implies that the clusters are distinct and well-defined. Generally, it is a very good model.

4.5 FINDINGS DISCUSSION

From the data mining aspect, it was discovered that the borough of Kensington and Chelsea had the most people affected by fire incidents caused by white goods with the most injuries and deaths. Croydon was in the top 5 of the number of incidents and the number of injuries in fire incidents within the period considered. When it comes to manufacturers of white goods, Bosch was the manufacturer that had the most white goods causing fires. Lastly, Tumble Dryer-Standard had the largest share of ignition sources of white goods.

For data prediction using supervised and unsupervised models, the Random Forest Classifier had an accuracy of 0.9918, a recall of 1, and a precision of 0.98. The model is 99% percent accurate when predicting and had a 100% chance of not predicting a false positive which means that all its positives are positive and a 98% chance of not predicting a false negative. The model looks perfect because it is preferable to have a higher value for recall than precision as human lives are involved.

For the unsupervised model, The K-means had a good silhouette score of 0.7078, and the clusters are well separated from each other. It is a very good

model with a very low WCSS of 40 which means the compactness and tightness of the clusters make it very good for discovering patterns.

The supervised model (Random Forest Classifier) showed low performance due to data imbalance. Under-sampling helped solve this as there was heavy bias because of the sheer large size of the majority class 0. The presence of irrelevant data in the dataset introduced bias into the Random Forest model and the K-means. K-means clustering also showed low performance because optimal hyperparameters were not picked.

The quality of this research can not be investigated as there is little to no known research on this part of fire prediction; the part that deals with human lives. There are not many existing academic works relating to the London Fire Brigade.

For the data mining process, the data was gotten from the London datastore, and a first data exploration was done. The data was cleaned by performing correlation analysis on the variables using a correlation matrix and chi-square test for categorical and numerical variables. Then a new target variable was generated from the injuries and deaths. All low-correlated and irrelevant variables were dropped. For the supervised modeling, One-Hot encoding was done on the data, and the model was under-sampled and then split into train and test data. It was fitted and predicted for the unsupervised modeling, Label Encoding was done on the data and then there was hyperparameters tunning followed by fitting the model and plotting the graphs.

5.0 CONCLUSION AND PERSPECTIVES

The Random Classifier can correctly predict if there is going to be a death or injury 99.18% of the time. Also, it has a higher accuracy of predicting a no-injury/death situation than an injury/death situation (The case of recall and precision).

The K-means algorithm can help to use pre-existing data to look for trends in the data that can cause injury/death like the Ignition Source and the location the fire started.

During the mining process, there were problems with the quality of the data like missing data, inconsistent formatting, and outliers. Data cleaning was very time-consuming during the data pre-processing. There was a problem with overfitting because of the selection of the wrong models.

In the future, there may be refining of predictive models. This will involve incorporating more extensive datasets and advanced machine learning algorithms which can improve the accuracy of predictions. There may also be collaborative data sharing which involves sharing data among relevant stakeholders and decision-makers.

Due to the complexity of my dataset, I had to change the topic from “LONDON FIRE BRIGADE: DISCOVERING EXCITING TRENDS OF THE MOBILIZATION RECORDS USING DATA MINING TECHNIQUES”. The dataset had about 489,000 data points and Google Colab could not handle the explanatory and prediction part because of the computational power involved.

5.1 RECOMMENDATIONS

There should be expansive collaborative partnerships with appliance manufacturers, and industry experts to learn more about the safety of white goods as this can lead to the development of improved safety standards. Strengthening the regulatory framework is also paramount as it can involve reviewing and strengthening existing safety regulations about white goods. Implementing real-time monitoring systems is also recommended to detect potential white goods with potential fire risks.

.

REFERENCES

[1] The Fire Brigades Union, “Why emergency response times matter to firefighters and the public,” [Online]. Available: https://www.epsu.org/sites/default/files/article/files/6367-Its-about-time-LOW-RES2.pdf. [Accessed 19 April 2023].

[2] J. R. Asor, J. L. Lerios, S. B. S. J. O. Padallan and C. A. C. Buama, “Fire incidents visualization and pattern recognition using machine learning algorithms,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, no. 3, pp. 1427 -1435, 2021.

[3] R. Rishickesh, A. Shahina and A. N. Khan, “Predicting Forest Fires using Supervised and Ensemble Machine Learning Algorithms,” *International Journal of Recent Technology and Engineering*, vol. 8, no. 2, pp. 2277-3878 , 2019.

[4] A. Jaafari, E. K. Zenner and B. T. Pham, “Wildfire spatial pattern analysis in the Zagros Mountains, Iran: A comparative study of decision tree based classifiers,” *Ecological Informatics*, vol. 43, pp. 200-211, 2018.

[5] B. T. Pham, A. Jaafari, M. Avand, N. Al-Ansari, T. D. Du, H. P. H. Yen, T. V. Phong, D. H. Nguyen, H. V. Le, D. Mafi-Gholami, I. Prakash, H. T. Thuy and T. T. Tuyen, “Performance Evaluation of Machine Learning Methods for Forest Fire Modeling and Prediction,” *Symmetry*, vol. 12, no. 6, p. 1022, 2020.

[6] M.-Y. Choi and S. Jun, “Fire Risk Assessment Models Using Statistical Machine Learning and Optimized Risk Indexing,” *MDPI*, vol. 10, no. 12, p. 4199, 2020.

APPENDICES

APPENDIX A

Load the Variables

```
[140] import pandas as pd
      import matplotlib.pyplot as plt
      %matplotlib inline
      import numpy as np
      import seaborn as sns
      #import the data into the csv
      df = pd.read_csv('/content/Fires in white goods from 2009.csv')
```

Total Number of Entries

```
[141] # Get the total count of entries
      total_entries = df.shape[0]

      print("Total entries:", total_entries)
```

```
Total entries: 4437
```

Missing Columns

```
[142] #replaces the blank entries with NaN
      df.replace(' ', np.nan, inplace=True)

      #Prints the number of columns for each attribute that is missing
      print(df.isnull().sum())
```

Data Exploration

+ Code + Text

```
[4]  # Histogram of Incidents by Year
     #Uses the histplot of the Seaborn Library
     vf = sns.histplot(df['Month'])
     x_labels = vf.get_xticklabels()
     vf.set_xticklabels(x_labels, rotation=45, ha="right")
```

Histogram of Ignition Source

```
[25] # Histogram of IncGeo_BoroughName
     vf = sns.histplot(df['IgnitionSource'])
     x_labels = vf.get_xticklabels()
     vf.set_xticklabels(x_labels, rotation=45, ha="right")
```

Pie Chart of Appliance Manufacturer

```
[26] #Count the occurrences of each value in the column and select the top 5 values
     top_values = df['ApplianceManufacturer'].value_counts().head(5)

     #Create a new DataFrame to store the top 5 values and their counts
     top_values_df = pd.DataFrame({'Values': top_values.index, 'Count': top_values.values})

     #Calculate the sum of counts for the remaining values
     other_count = df['ApplianceManufacturer'].value_counts().sum() - top_values_df['Count'].sum()

     #Create a new row in the DataFrame for the "Others" category
     top_values_df = top_values_df.append({'Values': 'All Others', 'Count': other_count}, ignore_index=True)

     #Create a pie chart using matplotlib
     plt.pie(top_values_df['Count'], labels=top_values_df['Values'], autopct='%1.1f%%')
     plt.axis('equal')
     plt.title('White goods by Top 5 Appliance Manufacturer')
     plt.show()
```

Create a new column named "NumPeopleAffected" from "NumFireDeaths" and "NumAllFireInjuries"

```
[145] import pandas as pd

      # Assuming you have a DataFrame named 'df' containing the fire incident data

      # Check if either NumFireDeaths or NumAllFireInjuries is more than 0
      df['NumPeopleAffected'] = 0
      df.loc[(df['NumFireDeaths'] > 0) | (df['NumAllFireInjuries'] > 0), 'NumPeopleAffected'] = 1
```

Reorder the Columns

```
[146] # Reorder the columns to have 'NumPeopleAffected' beside 'NumAllFireInjuries'
      df = df[['Year', 'Month', 'IncType', 'ParentPropertyType', 'NumFireDeaths',
               'NumAllFireInjuries', 'NumPeopleAffected', 'IncGeo_BoroughCode',
               'IncGeo_BoroughName', 'IncGeo_WardCode', 'IncGeo_WardName',
               'IgnitionSourcePower', 'IgnitionSource', 'ItemFirstIgnited',
               'LocationFireStarted', 'ApplianceManufacturer',
               'ApplianceManufacturerOther', 'MainCauseModel']]

      # Print the updated DataFrame
      print(df)
```

Calculate Correlation

```
[148] target_corr = df.corr()["NumPeopleAffected"]
      target_corr = target_corr.drop("NumPeopleAffected")

      fig, ax = plt.subplots(figsize=(10, 8))
      cmap = sns.diverging_palette(220, 10, as_cmap=True)

      sns.heatmap(target_corr.to_frame(), cmap=cmap, center=0, square=True, linewidths=.9, annot=True, cbar_kws={"shrink": .5})

      plt.title("Correlation with NumPeopleAffected")
      plt.show()
```

Chi-Squared Test

```
from scipy.stats import chi2_contingency
# Calculate the chi-square test for categorical variables
cat_vars = ['Month', 'IncType', 'IncGeo_BoroughName', 'ParentPropertyType','IncGeo_WardName', 'IncGeo_WardCode',
            'IncGeo_BoroughCode', 'IgnitionSourcePower', 'IgnitionSource',
            'ItemFirstIgnited', 'LocationFireStarted', 'ApplianceManufacturer',
            'MainCauseModel']
chi2_values = []
for var in cat_vars:
    contingency_table = pd.crosstab(df[var], df['NumPeopleAffected'])
    chi2, p, dof, expected = chi2_contingency(contingency_table)
    chi2_values.append(chi2)

# Print the chi-square test results
print(pd.DataFrame({'Variable': cat_vars, 'Chi-Square': chi2_values}))
```

Remove Unwanted Columns

```
[150] del_columns = ['Year','IncType', 'IgnitionSourcePower', 'Month','ApplianceManufacturerOther',
                     'IncGeo_WardName', 'IncGeo_WardCode','IncGeo_BoroughCode', 'MainCauseModel']

      df = df.drop(del_columns, axis=1)
      df.head()
```

	ParentPropertyType	NumFireDeaths	NumAllFireInjuries	NumPeopleAffected	IncGeo_BoroughName
0	Purpose Built Flats/Maisonettes	0	0	0	Lambeth
1	Purpose Built Flats/Maisonettes	0	0	0	Camden
2	Purpose Built Flats/Maisonettes	0	1	1	Havering
3	Purpose Built Flats/Maisonettes	0	0	0	Hammersmith and Fulham
4	Purpose Built Flats/Maisonettes	0	0	0	Wandsworth

Replace Blank with 'Other' in ApplianceManufacturer

```
[155] # Assuming you have a DataFrame called 'df' containing your data
      df['ApplianceManufacturer'] = df['ApplianceManufacturer'].fillna('Other')
```

Replace "Null" with 'Other' in ApplianceManufacturer

```
[156] # Assuming you have a DataFrame called 'df' containing your data
      df['ApplianceManufacturer'] = df['ApplianceManufacturer'].replace('Null', 'Other')
```

Borough Name value count

```
[157] print(df['IncGeo_BoroughName'].value_counts())
```

Changing Case

```
[158] df['IncGeo_BoroughName'] = df['IncGeo_BoroughName'].str.title()
```

Borough name value count to confirm

```
[159] print(df['IncGeo_BoroughName'].value_counts())
```

Borough counts by death

```
[242] # Group the dataframe by 'IncGeo_BoroughName' and calculate the sum of 'NumFireDeaths' for each borough
      borough_deaths = df.groupby('IncGeo_BoroughName')['NumFireDeaths'].sum()

      # Sort the borough_deaths series in descending order
      borough_deaths = borough_deaths.sort_values(ascending=False)

      # Select the top 5 boroughs and sum the remaining boroughs as 'Others'
      top_5 = borough_deaths[:5]
      others = borough_deaths[5:].sum()

      # Create a new series combining the top 5 and 'Others'
      combined = pd.concat([top_5, pd.Series({'Others': others})])

      # Plot the pie chart
      plt.pie(combined, labels=combined.index, autopct='%1.1f%%')
      plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
      plt.title('Fire Deaths by Borough (Top 5 + Others)')

      # Display the chart
      plt.show()
```

Plot the number of injuries by borough

```
# Group the dataframe by 'IncGeo_BoroughName' and calculate the sum of 'NumAllFireInjuries' for each borough
borough_injuries = df.groupby('IncGeo_BoroughName')['NumAllFireInjuries'].sum()

# Sort the borough_injuries series in descending order
borough_injuries = borough_injuries.sort_values(ascending=False)

# Select the top 5 boroughs and sum the remaining boroughs as 'Others'
top_5 = borough_injuries[:5]
others = borough_injuries[5:].sum()

# Create a new series combining the top 5 and 'Others'
combined = pd.concat([top_5, pd.Series({'Others': others})])

# Plot the pie chart
plt.pie(combined, labels=combined.index, autopct='%1.1f%%')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
plt.title('Fire Injuries by Borough (Top 5 + Others)')

# Display the chart
plt.show()
```

Histogram of Boroughs

```
# Histogram of IncGeo_BoroughName
vf = sns.histplot(df['IncGeo_BoroughName'])
x_labels = vf.get_xticklabels()
vf.set_xticklabels(x_labels, rotation=45, ha="right")
```

One Hot Encoding

Loading...

```
# Select the columns to be one-hot encoded
columns_to_encode = ['ParentPropertyType', 'IncGeo_BoroughName', 'IgnitionSource',
                     'ItemFirstIgnited', 'LocationFireStarted', 'ApplianceManufacturer']

# Perform one-hot encoding using Pandas get_dummies() method
encoded_columns = pd.get_dummies(df[columns_to_encode], prefix=columns_to_encode)

# Drop the original columns from the DataFrame
df = df.drop(columns_to_encode, axis=1)

# Concatenate the original DataFrame with the one-hot encoded columns
df_encoded = pd.concat([df, encoded_columns], axis=1)
```

APPENDIX B

Import the Libraries

```
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import seaborn as sns
from imblearn.under_sampling import RandomUnderSampler
from sklearn.utils import resample
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, roc_curve
```

Read the csv

```
[66] df = pd.read_csv('final_clean.csv')
```

Undersample the data

```
[62] # Separate the minority and majority classes
     minority_class = df[df['NumPeopleAffected'] == 1]
     majority_class = df[df['NumPeopleAffected'] == 0]

     # Undersample the majority class
     undersampled_majority = resample(majority_class,
                                      replace=False,  # Set to False for undersampling
                                      n_samples=len(minority_class),  # Match the number of minority samples
                                      random_state=42)  # Set a random seed for reproducibility

     # Combine the minority class and undersampled majority class
     undersampled_df = pd.concat([undersampled_majority, minority_class])

     # Split the undersampled dataset into features (X) and target variable (y)
     X = undersampled_df.drop('NumPeopleAffected', axis=1)
```

✓ 0s completed at 18:47

```
[62]
     # Split the undersampled dataset into features (X) and target variable (y)
     X = undersampled_df.drop('NumPeopleAffected', axis=1)
     y = undersampled_df['NumPeopleAffected']
```

Split into training and testing datasets

```
[73] # Split the data into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
```

```
0s
     # Create a Random Forest Classifier
     rf_classifier = RandomForestClassifier(random_state=42)

     # Fit the model to the training data
     rf_classifier.fit(X_train, y_train)

     # Make predictions on the test set
     y_pred = rf_classifier.predict(X_test)

     #AUC and ROC
     rfc_auc = roc_auc_score(y_test, y_pred)
     rfc_fpr, rfc_tpr, _ = roc_curve(y_test, y_pred)

     # Evaluate the model's performance
     accuracy = accuracy_score(y_test, y_pred)
     precision = precision_score(y_test, y_pred)
     recall = recall_score(y_test, y_pred)
     f1 = f1_score(y_test, y_pred)

     print("Accuracy:", accuracy)
     print("Precision:", precision)
     print("Recall:", recall)
     print("F1-score:", f1)
```

```
Accuracy: 0.9918032786885246
Precision: 0.98
```

Classification Report

```
[75] # Generate the classification report
     report = classification_report(y_test, y_pred)

     print("Classification Report:")
     print(report)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99        73
           1       0.98      1.00      0.99        49

    accuracy                           0.99       122
   macro avg       0.99      0.99      0.99       122
weighted avg       0.99      0.99      0.99       122
```

Confusion Matrix

```
[44] # Create a confusion matrix
     cm = confusion_matrix(y_test, y_pred)

     # Create a heatmap of the confusion matrix
     plt.figure(figsize=(8, 6))
     sns.heatmap(cm, annot=True, cmap="Blues", fmt="d", cbar=False)
     plt.title('Confusion Matrix')
     plt.xlabel('Predicted Labels')
     plt.ylabel('True Labels')
     plt.show()
```

The ROC Curve

```
[36] # Plot the ROC curves
     plt.figure(figsize=(8, 6))
     plt.plot(rfc_fpr, rfc_tpr, label='Random Forest (AUC = {:.3f})'.format(rfc_auc))
     plt.plot([0, 1], [0, 1], linestyle='--', color='grey')
     plt.xlabel('False Positive Rate')
     plt.ylabel('True Positive Rate')
     plt.title('Receiver Operating Characteristic (ROC) Curve')
     plt.legend()
     plt.show()
```

Unsupervised Model

Import Libraries

```
[40] import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn.cluster import KMeans
     from sklearn.metrics import silhouette_score
     from sklearn.preprocessing import MinMaxScaler
     import matplotlib.pyplot as plt
     from mpl_toolkits.mplot3d import Axes3D
     from sklearn.preprocessing import LabelEncoder
```

Label Encode the Categorical Variables

```
[43] # Select the columns to be label encoded
     columns_to_encode = ['ParentPropertyType', 'IncGeo_BoroughName', 'IgnitionSource',
                          'ItemFirstIgnited', 'LocationFireStarted', 'ApplianceManufacturer']

     # Create a label encoder object
     label_encoder = LabelEncoder()

     # Iterate over the selected columns and perform label encoding
     for column in columns_to_encode:
         # Fit the label encoder on the column's values and transform the column
         df[column] = label_encoder.fit_transform(df[column])

     # Save the modified DataFrame back to a CSV file
     df.to_csv('encoded_file.csv', index=False)
```

Implement the Clusters

```
# Selecting the columns
columns = ['IgnitionSource', 'LocationFireStarted','NumPeopleAffected']

# Preprocess the data by scaling the features
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(df[columns])

# Determine the optimal number of clusters using the elbow method
distortions = []
silhouette_scores = []
for i in range(4, 11):
    kmeans = KMeans(n_clusters=i, random_state=0)
    kmeans.fit(data_scaled)
    distortions.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(data_scaled, kmeans.labels_))

# Based on the elbow curve and silhouette score, choose the appropriate number of clusters
num_clusters = 10

# Apply K-means clustering with the chosen number of clusters
kmeans = KMeans(n_clusters=num_clusters, random_state=0)
kmeans.fit(data_scaled)

# Add the cluster labels to the original DataFrame
df['Cluster'] = kmeans.labels_
silhouette_avg = silhouette_score(data_scaled, kmeans.labels_)

# Print the resulting clusters
print(df[['NumPeopleAffected', 'Cluster']])
print(f"Silhouette Score: {silhouette_avg}")
```

Elbow Curve

```
[46] # Plot the elbow curve to identify the optimal number of clusters
     plt.figure(figsize=(12, 4))
     plt.subplot(1, 2, 1)
     plt.plot(range(4, 11), distortions, marker='o')
     plt.xlabel('Number of Clusters')
     plt.ylabel('WCSS')
     plt.title('Elbow Curve')
```

Silhouette Score Plot

```
[47] plt.subplot(1, 2, 2)
     plt.plot(range(4, 11), silhouette_scores, marker='o')
     plt.xlabel('Number of Clusters')
     plt.ylabel('Silhouette Score')
     plt.title('Silhouette Score')
     plt.tight_layout()
     plt.show()
```

Cluster Plot

```
[48] # Perform a 3D cluster plot using three axes of the dataset
     fig = plt.figure(figsize=(8, 8))
     ax = fig.add_subplot(111, projection='3d')

     # Cluster centers
     cluster_centers = scaler.inverse_transform(kmeans.cluster_centers_)
     ax.scatter(cluster_centers[:, 0], cluster_centers[:, 1], cluster_centers[:, 2], c='red', s=200, marker='X', label='Cluster Centers')

     ax.scatter(df['IgnitionSource'], df['LocationFireStarted'], df['NumPeopleAffected'], c=df['Cluster'])

     ax.set_xlabel('IgnitionSource')
     ax.set_ylabel('LocationFireStarted')
     ax.set_zlabel('NumPeopleAffected')
     ax.set_title('Cluster Plot')
```

Link of Colab: https://colab.research.google.com/drive/1fGpyzWki4HhzSPhKS3IkZi7qi4laHOak?usp=sharing.