Earthquake Magnitude, Damage and Impact

Ankit Tamrakar
University of Wolverhampton
Herald College Kathmandu
Kathmandu, Nepal
2050058

Bibek Shrestha
University of Wolverhampton
Herald College Kathmandu
Kathmandu, Nepal
2049851

In 2015, Nepal was struck by a devastating earthquake that impacted millions of people living in 11 of the nation's 77 districts. Inside this notebook, we attempt to examine data generated by the Nepalese govt, that also conducted an enormous survey questionnaire utilizing digital technology to evaluate structural damage in earthquake-affected districts for both the means of furthering aid. This same information obtained, but at the other hand, is indeed not restricted to constructing end up making and damage data; rather, it's indeed rich to economic and educational information of the people who depend, who we will aim to determine utilizing Python to learn further about Nepalese culture as well as how the earthquake impacted their lives.

This information was gathered during January to May of 2016 and therefore is freely accessible via the Nepal Earthquake: Open Data Portal.

I. Background of the Study

The 7.8-magnitude Gorkha earthquake struck Nepal in April of 2015. It struck near the Gorkha district of Gandaki Pradesh. Almost 9,000 lives were lost, millions of people were displaced, and $10 billion in damages were incurred—roughly half of Nepal's nominal GDP. In the years since, the Nepalese government has worked tirelessly to assist in the reconstruction of the infrastructure in the impacted districts. The National Planning Commission, in partnership with Kathmandu Living Labs and the Central Bureau of Statistics, has developed a plan for the city of Kathmandu, has produced a number of documents during this process.

Among the most extensive post-disaster databases ever compiled, containing pertinent data on earthquake damages, household conditions, including socioeconomic statistics.

The volume of information in the database is on the structure and legal ownership of the buildings. Every row in the collection represents a single earthquake-damaged structure in the impacted area.

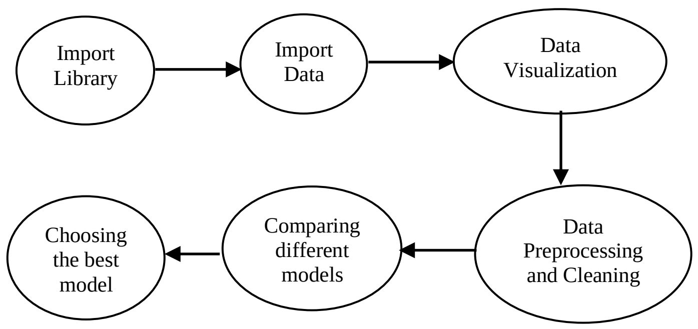
The structure id column serves as a great and random identifier in this dataset, which has 39 columns The remaining 38 features are covered in the next section. Random lowercase ascii characters have been used to disguise categorical variables. The presence of the same character in different columns does not mean that the original value was the same.

II. Related Work

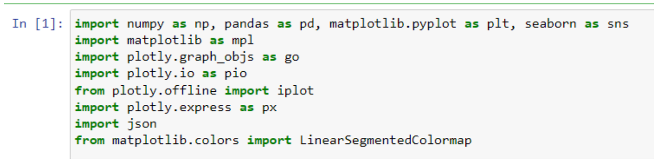
The findings of earthquake damage prediction to structures are often key parameters in earthquake damage prediction theory, according to "Rapid Prediction for Earthquake Damage to Structures based on Fuzzy Analysis." Whereas the concept of earthquake damage predicting for structures may involve explanations of certain impact components and judgment indexes, the prediction results are often state variables. Moreover, the descriptions of a number of impact elements and judgment indices are vague. As a consequence, a membership function for forecasting earthquake damage to buildings is created based on fuzzy mathematics, using seismic vulnerability indices as measurements for earthquake damage, mean earthquake damage indices as returning indicator, & factors causing of buildings as alteration index. Because there are no impact factors, a modified coefficient function is created utilizing representative sample and a proximity function. [12]

Earthquake Prediction Using Map Reduce Framework addresses a revolutionary way using the Map Reduce model, to developed and predict the next earthquake from tons of international geological survey data. Furthermore, the Map and Reduce function is used to determine the most shaking location, the location closest to the fault line, and the present location shakes per minute. Aside from the functionalities listed above, a separate Map and Reduce function has been created to examine the sheer number of earthquakes per day. The final result reveals which place experienced the most tremors and which location is the most likely to have an earthquake. [11]

III. Methodology

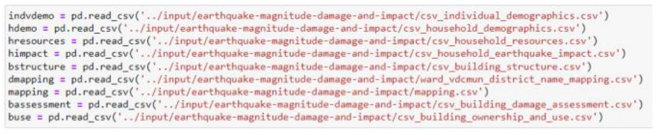


A. Importing Library



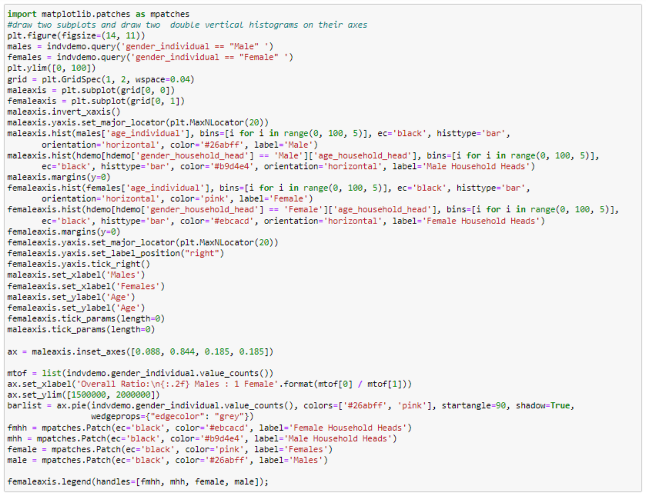
```
In [1]: import numpy as np, pandas as pd, matplotlib.pyplot as plt, seaborn as sns
        import matplotlib as mpl
        import plotly.graph_objs as go
        import plotly.io as pio
        from plotly.offline import iplot
        import plotly.express as px
        import json
        from matplotlib.colors import LinearSegmentedColormap
```

B. Importing Data

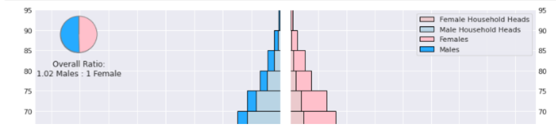


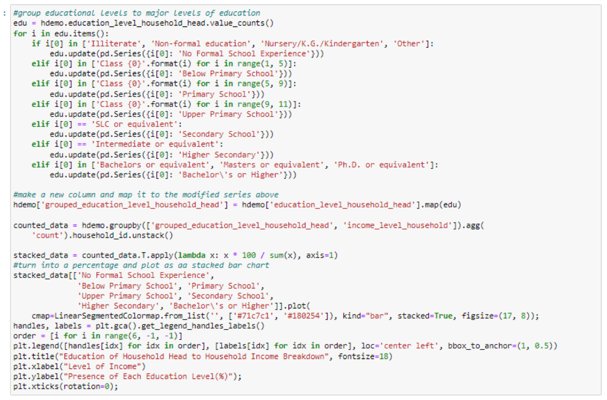
```
indvdemo = pd.read_csv('../input/earthquake-magnitude-damage-and-impact/csv_individual_demographics.csv')
hdemo = pd.read_csv('../input/earthquake-magnitude-damage-and-impact/csv_household_demographics.csv')
hresources = pd.read_csv('../input/earthquake-magnitude-damage-and-impact/csv_household_resources.csv')
himpact = pd.read_csv('../input/earthquake-magnitude-damage-and-impact/csv_household_earthquake_impact.csv')
bstructure = pd.read_csv('../input/earthquake-magnitude-damage-and-impact/csv_building_structure.csv')
dmapping = pd.read_csv('../input/earthquake-magnitude-damage-and-impact/ward_vdcmun_district_name_mapping.csv')
mapping = pd.read_csv('../input/earthquake-magnitude-damage-and-impact/mapping.csv')
bassessment = pd.read_csv('../input/earthquake-magnitude-damage-and-impact/csv_building_damage_assessment.csv')
buse = pd.read_csv('../input/earthquake-magnitude-damage-and-impact/csv_building_ownership_and_use.csv')
```

C. Data Visualization

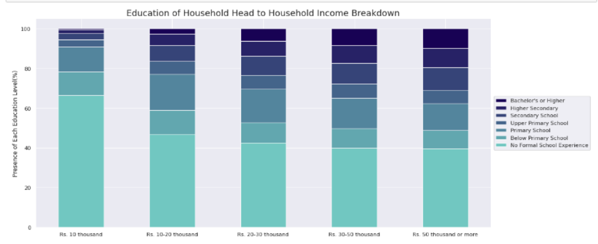


```
import matplotlib.patches as mpatches
#draw two subplots and draw two  double vertical histograms on their axes
plt.figure(figsize=(14, 11))
males = indvdemo.query('gender_individual == "Male" ')
females = indvdemo.query('gender_individual == "Female" ')
plt.ylim([0, 100])
grid = plt.GridSpec(1, 2, wspace=0.04)
maleaxis = plt.subplot(grid[0, 0])
femaleaxis = plt.subplot(grid[0, 1])
maleaxis.invert_xaxis()
maleaxis.yaxis.set_major_locator(plt.MaxNLocator(20))
maleaxis.hist(males['age_individual'], bins=[i for i in range(0, 100, 5)], ec='black', histtype='bar',
       orientation='horizontal', color='#26abff', label='Male')
maleaxis.hist(hdemo[hdemo['gender_household_head'] == 'Male']['age_household_head'], bins=[i for i in range(0, 100, 5)],
       ec='black', histtype='bar', color='#b9d4e4', orientation='horizontal', label='Male Household Heads')
maleaxis.margins(y=0)
femaleaxis.hist(females['age_individual'], bins=[i for i in range(0, 100, 5)], ec='black', histtype='bar',
       orientation='horizontal', color='pink', label='Female')
femaleaxis.hist(hdemo[hdemo['gender_household_head'] == 'Female']['age_household_head'], bins=[i for i in range(0, 100, 5)],
       ec='black', histtype='bar', color='#ebcacd', orientation='horizontal', label='Female Household Heads')
femaleaxis.margins(y=0)
femaleaxis.yaxis.set_major_locator(plt.MaxNLocator(20))
femaleaxis.yaxis.set_label_position("right")
femaleaxis.yaxis.tick_right()
maleaxis.set_xlabel('Males')
femaleaxis.set_xlabel('Females')
maleaxis.set_ylabel('Age')
femaleaxis.set_ylabel('Age')
femaleaxis.tick_params(length=0)
maleaxis.tick_params(length=0)

ax = maleaxis.inset_axes([0.088, 0.844, 0.185, 0.185])

mtof = list(indvdemo.gender_individual.value_counts())
ax.set_xlabel('Overall Ratio:\n{:.2f} Males : 1 Female'.format(mtof[0] / mtof[1]))
ax.set_ylim([1500000, 2000000])
barlist = ax.pie(indvdemo.gender_individual.value_counts(), colors=['#26abff', 'pink'], startangle=90, shadow=True,
                 wedgeprops={"edgecolor": "grey"})
fmhh = mpatches.Patch(ec='black', color='#ebcacd', label='Female Household Heads')
mhh = mpatches.Patch(ec='black', color='#b9d4e4', label='Male Household Heads')
female = mpatches.Patch(ec='black', color='pink', label='Females')
male = mpatches.Patch(ec='black', color='#26abff', label='Males')

femaleaxis.legend(handles=[fmhh, mhh, female, male]);
```

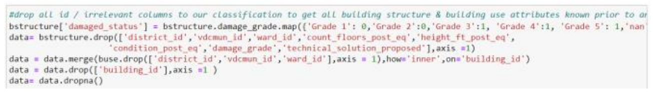




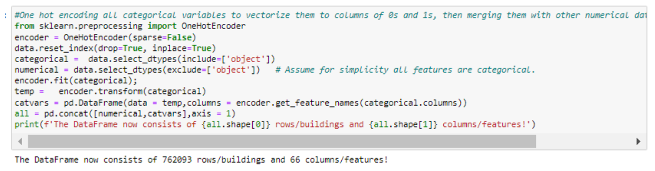
```
: #group educational levels to major levels of education
  edu = hdemo.education_level_household_head.value_counts()
  for i in edu.items():
      if i[0] in ['Illiterate', 'Non-formal education', 'Nursery/K.G./Kindergarten', 'Other']:
          edu.update(pd.Series({i[0]: 'No Formal School Experience'}))
      elif i[0] in ['Class {0}'.format(i) for i in range(1, 5)]:
          edu.update(pd.Series({i[0]: 'Below Primary School'}))
      elif i[0] in ['Class {0}'.format(i) for i in range(5, 9)]:
          edu.update(pd.Series({i[0]: 'Primary School'}))
      elif i[0] in ['Class {0}'.format(i) for i in range(9, 11)]:
          edu.update(pd.Series({i[0]: 'Upper Primary School'}))
      elif i[0] == 'SLC or equivalent':
          edu.update(pd.Series({i[0]: 'Secondary School'}))
      elif i[0] == 'Intermediate or equivalent':
          edu.update(pd.Series({i[0]: 'Higher Secondary'}))
      elif i[0] in ['Bachelors or equivalent', 'Masters or equivalent', 'Ph.D. or equivalent']:
          edu.update(pd.Series({i[0]: 'Bachelor\'s or Higher'}))

  #make a new column and map it to the modified series above
  hdemo['grouped_education_level_household_head'] = hdemo['education_level_household_head'].map(edu)

  counted_data = hdemo.groupby(['grouped_education_level_household_head', 'income_level_household']).agg(
      'count').household_id.unstack()

  stacked_data = counted_data.T.apply(lambda x: x * 100 / sum(x), axis=1)
  #turn into a percentage and plot as aa stacked bar chart
  stacked_data[['No Formal School Experience',
                'Below Primary School', 'Primary School',
                'Upper Primary School', 'Secondary School',
                'Higher Secondary', 'Bachelor\'s or Higher']].plot(
      cmap=LinearSegmentedColormap.from_list('', ['#71c7c1', '#180254']), kind="bar", stacked=True, figsize=(17, 8));
  handles, labels = plt.gca().get_legend_handles_labels()
  order = [i for i in range(6, -1, -1)]
  plt.legend([handles[idx] for idx in order], [labels[idx] for idx in order], loc='center left', bbox_to_anchor=(1, 0.5))
  plt.title("Education of Household Head to Household Income Breakdown", fontsize=18)
  plt.xlabel("Level of Income")
  plt.ylabel("Presence of Each Education Level(%)");
  plt.xticks(rotation=0);
```



D. Data Preprocssing and Cleaning

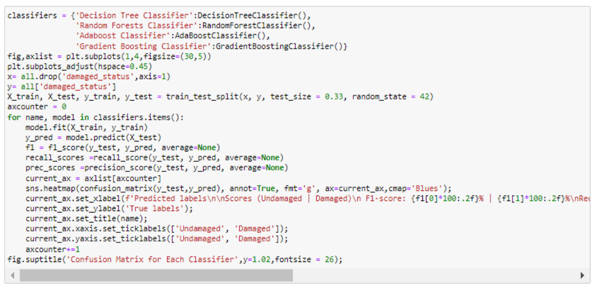


```
#drop all id / irrelevant columns to our classification to get all building structure & building use attributes known prior to ar
bstructure['damaged_status'] = bstructure.damage_grade.map({'Grade 1': 0,'Grade 2':0,'Grade 3':1, 'Grade 4':1, 'Grade 5': 1,'nan'
data= bstructure.drop(['district_id','vdcmun_id','ward_id','count_floors_post_eq','height_ft_post_eq',
                'condition_post_eq','damage_grade','technical_solution_proposed'],axis =1)
data = data.merge(buse.drop(['district_id','vdcmun_id','ward_id'],axis = 1),how='inner',on='building_id')
data = data.drop(['building_id'],axis =1 )
data= data.dropna()
```

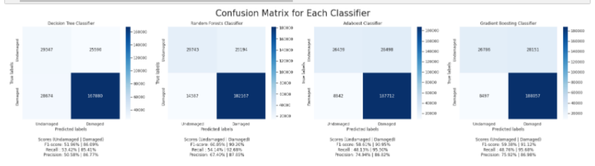


```
: #One hot encoding all categorical variables to vectorize them to columns of 0s and 1s, then merging them with other numerical dat
  from sklearn.preprocessing import OneHotEncoder
  encoder = OneHotEncoder(sparse=False)
  data.reset_index(drop=True, inplace=True)
  categorical =  data.select_dtypes(include=['object'])
  numerical = data.select_dtypes(exclude=['object'])   # Assume for simplicity all features are categorical.
  encoder.fit(categorical);
  temp =   encoder.transform(categorical)
  catvars = pd.DataFrame(data = temp,columns = encoder.get_feature_names(categorical.columns))
  all = pd.concat([numerical,catvars],axis = 1)
  print(f'The DataFrame now consists of {all.shape[0]} rows/buildings and {all.shape[1]} columns/features!')

  The DataFrame now consists of 762093 rows/buildings and 66 columns/features!
```

E. Comparing different models

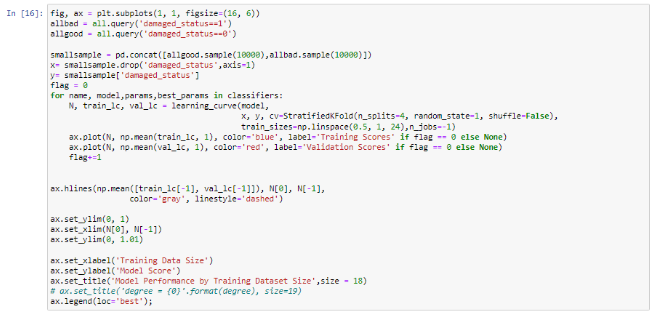


```
classifiers = {'Decision Tree Classifier':DecisionTreeClassifier(),
               'Random Forests Classifier':RandomForestClassifier(),
               'Adaboost Classifier':AdaBoostClassifier(),
               'Gradient Boosting Classifier':GradientBoostingClassifier()}
fig,axlist = plt.subplots(1,4,figsize=(30,5))
plt.subplots_adjust(hspace=0.45)
x= all.drop('damaged_status',axis=1)
y= all['damaged_status']
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.33, random_state = 42)
axcounter = 0
for name, model in classifiers.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    f1 = f1_score(y_test, y_pred, average=None)
    recall_scores =recall_score(y_test, y_pred, average=None)
    prec_scores =precision_score(y_test, y_pred, average=None)
    current_ax = axlist[axcounter]
    sns.heatmap(confusion_matrix(y_test,y_pred), annot=True, fmt='g', ax=current_ax,cmap='Blues');
    current_ax.set_xlabel(f'Predicted labels\n\nScores (Undamaged | Damaged)\n F1-score: {f1[0]*100:.2f}% | {f1[1]*100:.2f}%\nRec
    current_ax.set_ylabel('True labels');
    current_ax.set_title(name);
    current_ax.xaxis.set_ticklabels(['Undamaged', 'Damaged']);
    current_ax.yaxis.set_ticklabels(['Undamaged', 'Damaged']);
    axcounter+=1
fig.suptitle('Confusion Matrix for Each Classifier',y=1.02,fontsize = 26);
```

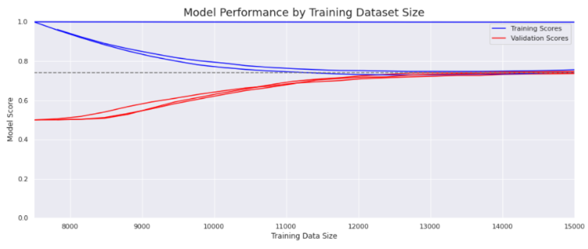


```
Best parameters for Gradient Boosting Classifier are: {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 250}
Best parameters for Random Forests Classifier are: {'criterion': 'entropy', 'max_depth': 9, 'max_features': 'auto', 'n_estimato
rs': 100}
Best parameters for Adaboost Classifier are: {'learning_rate': 1, 'n_estimators': 50}
```

F. Choosing the best model

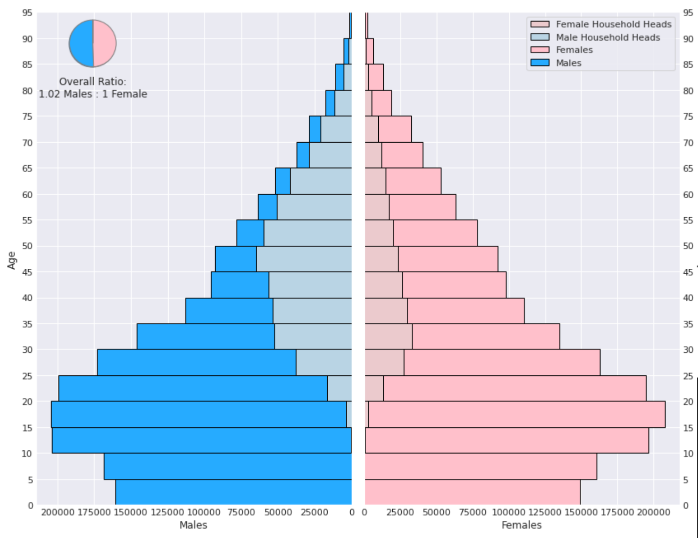


```
In [16]: fig, ax = plt.subplots(1, 1, figsize=(16, 6))
         allbad = all.query('damaged_status==1')
         allgood = all.query('damaged_status==0')

         smallsample = pd.concat([allgood.sample(10000),allbad.sample(10000)])
         x= smallsample.drop('damaged_status',axis=1)
         y= smallsample['damaged_status']
         flag = 0
         for name, model,params,best_params in classifiers:
             N, train_lc, val_lc = learning_curve(model,
                                                  x, y, cv=StratifiedKFold(n_splits=4, random_state=1, shuffle=False),
                                                  train_sizes=np.linspace(0.5, 1, 24),n_jobs=-1)
             ax.plot(N, np.mean(train_lc, 1), color='blue', label='Training Scores' if flag == 0 else None)
             ax.plot(N, np.mean(val_lc, 1), color='red', label='Validation Scores' if flag == 0 else None)
             flag+=1


         ax.hlines(np.mean([train_lc[-1], val_lc[-1]]), N[0], N[-1],
                         color='gray', linestyle='dashed')

         ax.set_ylim(0, 1)
         ax.set_xlim(N[0], N[-1])
         ax.set_ylim(0, 1.01)

         ax.set_xlabel('Training Data Size')
         ax.set_ylabel('Model Score')
         ax.set_title('Model Performance by Training Dataset Size',size = 18)
         # ax.set_title('degree = {0}'.format(degree), size=19)
         ax.legend(loc='best');
```



IV. Result and Discussion

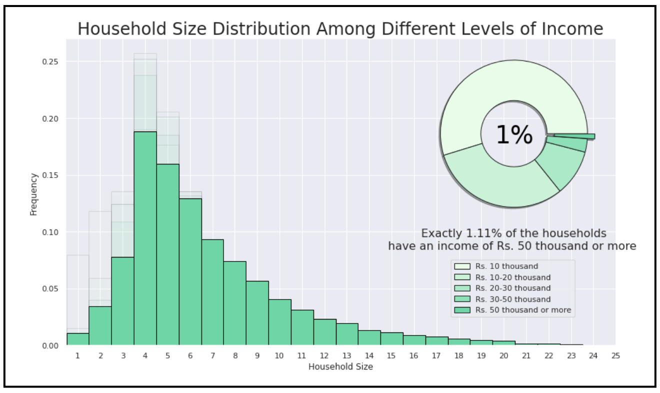
Beginning from demography, a sample data will assist us learn a lot further about Nepal's citizens, including age-sex ratio, including projected migration patterns.



This dataset's demographic pyramid strongly resembles a late-growing population pyramid, which means had been quickly increasing place at a single time before decreasing more in later generations. This might be result of a variety of factors such as economical and societal diversity, political uncertainty leads to reduced fertility rates and greater infant death rates, and etc.

So because male to female proportion with in overall population is about identical, there isn't really much of variance with in number of females to males across the age categories. Just at far elderly ages (75+) do we find how females constantly outweigh males, supporting the ancient adage that women has higher life expectancy than men.

Overall proportion of rural households with in demographic, as well as the major means of revenue for every home, is depicted in the chart figure. Throughout the individual age categories, male household heads are nearly



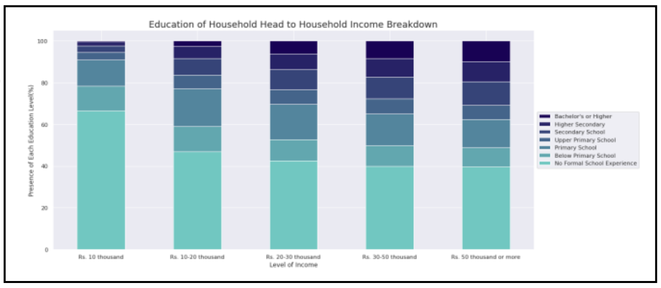
twice to quadruple the number of female household heads.

Each and every person inside the dataset is a member of ahome headed with one of the household.

As shown by the pie chart on the top right, 55% of the households have an income of Rs. 10 thousand or possibly less. This leaves an average of about 4-6 people, as suggested by the frequency histogram living on ($83.34 USD) per month assuming it's only source of income. In extreme cases, the number of people living in households with the lowest classification of income can reach up to 18 residents.

As you interact with the figure above, you will find that household size distribution in higher income households shifts more towards the right, meaning more people reside in the households with larger incomes on average. You should also notice that these higher income households make up less and less of the population the more they earn a month.

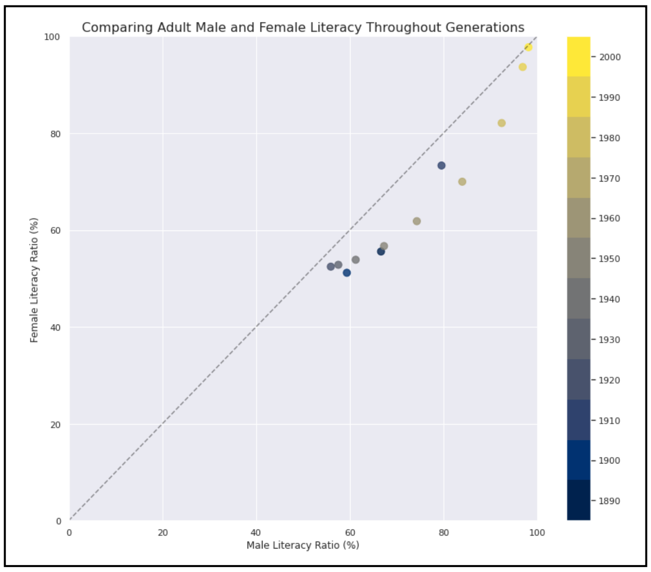
With the highest level of income making up only about 1% of the population, what does it take to earn this much for the household heads in our dataset at least in terms of education?



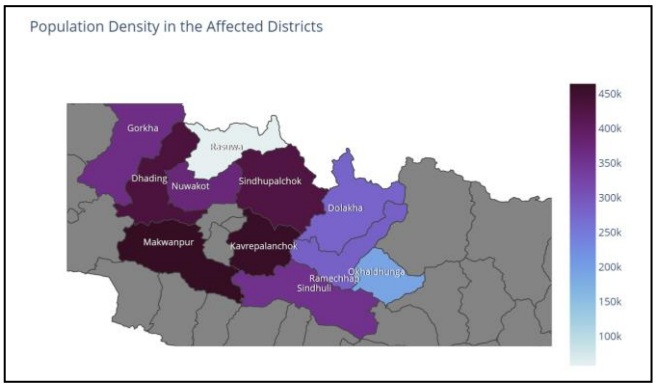
Having 40% upon those earning Rs. 50 thousands or even more (about $416.71 Dollars), this appears that to be uneducated in Nepal with none formal education doesn't really prevent someone from earning the highest monthly categorization in this dataset.

Inside the accompanying bar graphs, nevertheless, there is indeed a reasonably clear pattern indicating people with better and finer academic credentials grow in proportion as economy grows.

In each and every economic bracket, as seen in the chart image, at least 40% of family members are uneducated. Because we've been discussing of mature family members through the use of a greater mean lifespan than that of the majority of the people. What else can we say about the populations in the coming generations? Do they have a higher level of education than their forefathers?



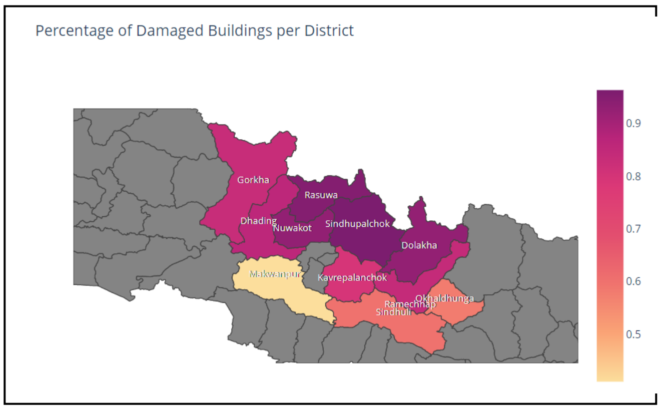
Nevertheless, it looks that the younger generations of the population will be on the correct path, since there seems to be a link between how lately a someone was born and the likelihood of being educated. This will only be excellent sign, with the literacy level improving not just nationally, as well as in the proportions among the genders reaching the horizontal equity lines, implying that both gender are similarly educated in subsequent years



Makwanpur and Kavrepalanchok appear to become the most densely inhabited regions of any and all, with approximately 1 million people affected by the earthquake in each. Having roughly 58 thousand persons, Rasuwa is the lowest populous district, with the rest of the districts falling somewhere in.

On another graphic, the uneven total population from among districts was depicted, which might lead to a misconception of the actual damage.

To correct this, the graph above depicts the proportion of total structures that required extensive repairs as a consequence of the earthquake. The third of five impact categorization grades accessible in our dataset.



Regrettably, the lowest inhabited sector was among the hardest hit. Throughout Gorkha through Okhaldhunga, there's also a distinct focus or "route" that the earthquake traced.

Sindhupalchok is indeed the district that has been the most badly affected, with nearly 96 percent of its buildings requiring extensive restoration.

Classification

We'll attempt to predict which buildings would've been seriously damaged after an earthquake of the same size using predictive methods, including which buildings really wouldn't. Prior to the crisis, we'll utilize all existing knowledge on the structures, include floor numbers, floor kind, building elevation and space, and much more.

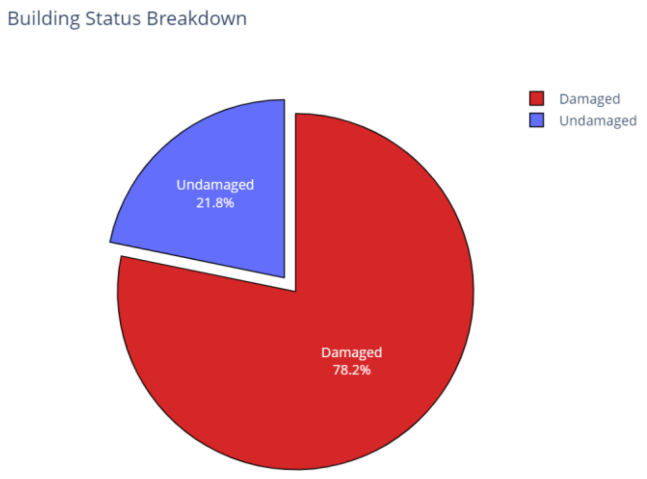
We'll put methods to the trial for this classifier that employ tree structure as forecasting models in some form or the other. We're going to put the following things to the test:

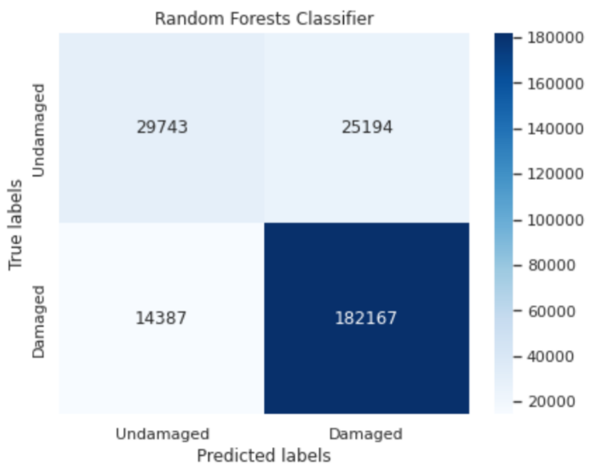
- Decision Tree Classifier
- AdaBoost Classifier
- Random Forest Classifier
- Gradient Boosting Classifier

and determine whichever one does the greatest job of categorizing the dataset

Then see how similar the categories in the dataset would be before we start categorizing. Every structure that has "Grade 3" or higher destruction will be considered destroyed, while others will be considered intact.

Now let us look at all of the structures in the dataset that have been harmed vs. those that have not. Every irrelevant column will be removed too though.





Scores (Undamaged | Damaged)
F1-score: 60.05% | 90.20%
Recall : 54.14% | 92.68%
Precision: 67.40% | 87.85%

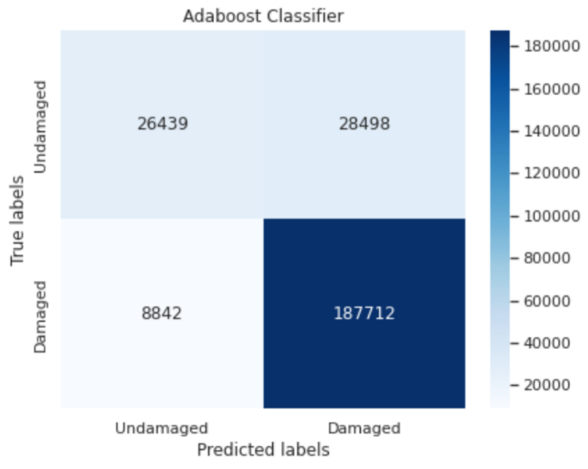
The dataset appears to be severely imbalanced. Sadly, barely 21.8 percent are unaffected. It might lead towards the simulation being heavily skewed. This must be considered while analyzing our models' indicators to prevent any bias when evaluating various categorization techniques.

We'll usually involving all category categories that used a OneHotEncoder to convert it into fields of 0s and 1s before providing any one of our information to the framework:

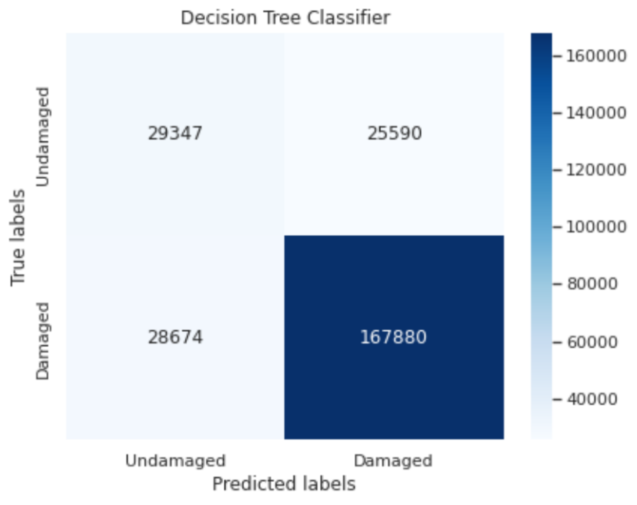
We'll immediately do an evaluation heatmap in order to comprehend the effectiveness. We'll retrain the algorithms with their configuration, which aren't likely to produce the highest performance, particularly because they're designed for precision above everything, that is not very useful. Because to the substantial unbalance in this instance.

It may, nevertheless, assist us in eliminating the design that is the most suitable.

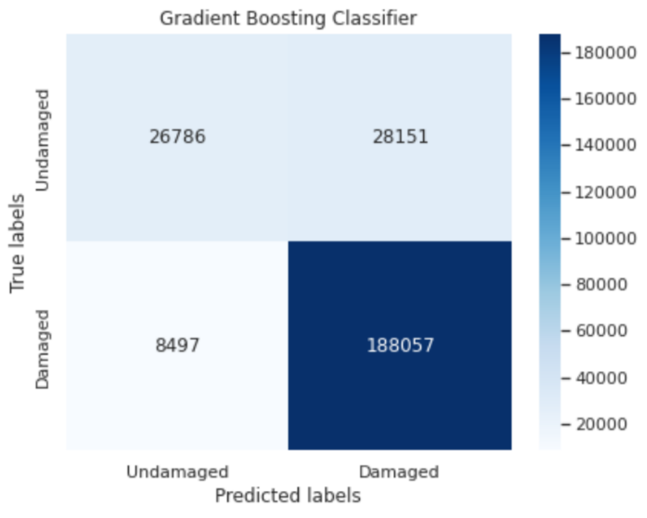
.



Scores (Undamaged | Damaged)
F1-score: 58.61% | 90.95%
Recall : 48.13% | 95.50%
Precision: 74.94% | 86.82%



Scores (Undamaged | Damaged)
F1-score: 51.96% | 86.09%
Recall : 53.42% | 85.41%
Precision: 50.58% | 86.77%



Scores (Undamaged | Damaged)
F1-score: 59.38% | 91.12%
Recall : 48.76% | 95.68%
Precision: 75.92% | 86.98%

This decision tree algorithm really does have the lowest efficiency, as illustrated by the heatmaps shown. This is to be anticipated, as alternative models use enhancers or a myriad of trees to accomplish the very same notion of a decision tree.

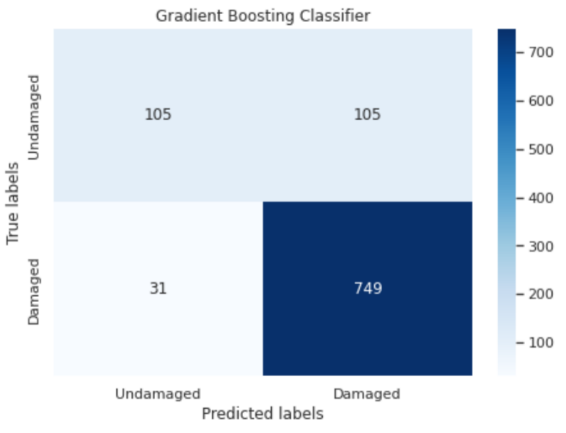
Even new alternatives, on the other hand, aren't any superior. The F1, recall, and accuracy ratings provided at the bottom of the each heatmap have always been greater for the damage category, that makes up the bulk of the heatmaps. The results can be quite deceiving because F1 is computed using accuracy and recall, which are both substantially skewed due to the data mismatch.

To put it another way, the model's ratings are exaggerated because of the large number of structures that were destroyed and then were labeled as such by the algorithm. This heatmap, which carries roughly the very same hue at the peak, and the significantly lower scores for damaged projections (true negatives) that verge on 50 percent recall/precision, both underscore this.

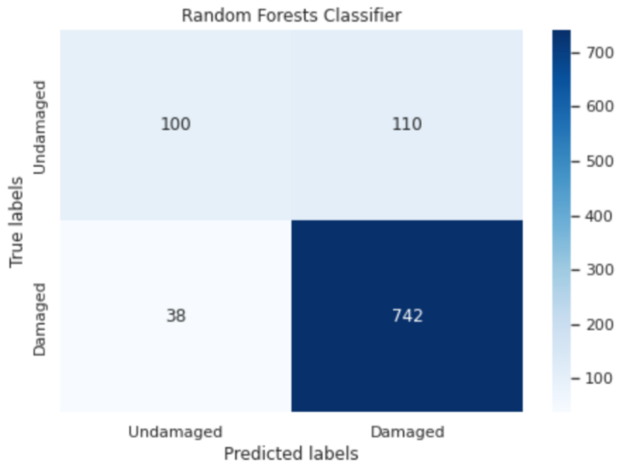
We must also highlight that in this use instance of a system, damaged building recall is meant to be the most relevant statistic, since recall reflects the number of buildings that were really damaged but forecasted to be undamaged. Usually greater the recall, the better the model, although this isn't always the case!

Although recall is critical, this is not the sole statistic to consider, since we may achieve 100% memory by labeling all structures as destroyed. A practical model should also have a modest level of accuracy, which this one lacks. The prediction of intact structures is left to chance in all standard hyper - parameter trained models with the this imbalanced data.

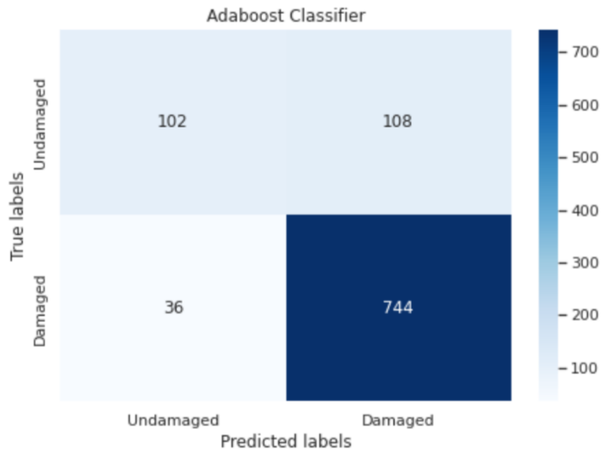
The disconcertingly high F1-score is inflated only by the exceptionally high precision - recall required to recognize the overwhelming amount of probable positives (damaged structures). This is something that you should avoid. The goal is to increase recall and precision while reducing false results as much as feasible. So now we've limited up our new capability, let's get started. Secondly, we'll target a specific subset of data by doing a merge look for the perfect model parameters from the previous three modeling in the hopes of getting good outcomes.



Scores (Undamaged | Damaged)
F1-score: 60.69% | 91.68%
Recall : 50.00% | 96.03%
Precision: 77.21% | 87.70%



Scores (Undamaged | Damaged)
F1-score: 57.47% | 90.93%
Recall : 47.62% | 95.13%
Precision: 72.46% | 87.09%

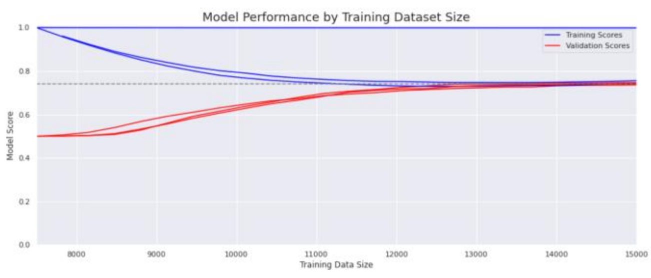


Scores (Undamaged | Damaged)
F1-score: 58.62% | 91.18%
Recall : 48.57% | 95.38%
Precision: 73.91% | 87.32%

Consequently, given an uneven dataset, our models appear to be doomed. Even exploring the hyper - parameter grid produces almost no improvement in the erroneous reasonable interest rate.

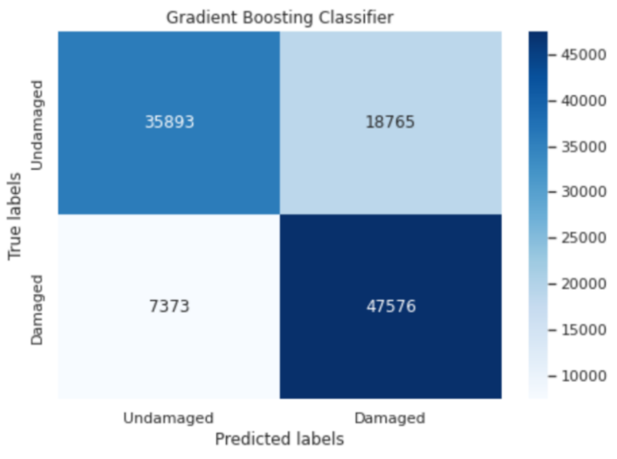
We'll strive for just an equivalent sampling of each category then evaluate our results under sampling to modify the hyper - parameters to retrain the model.

And save effort, we'll only utilize a part of the data on the grid search. To ensure that our grid search is optimum and produces great outcomes, we'll create a learning curve to observe when the classifiers' efficiency stops improving as data grows.

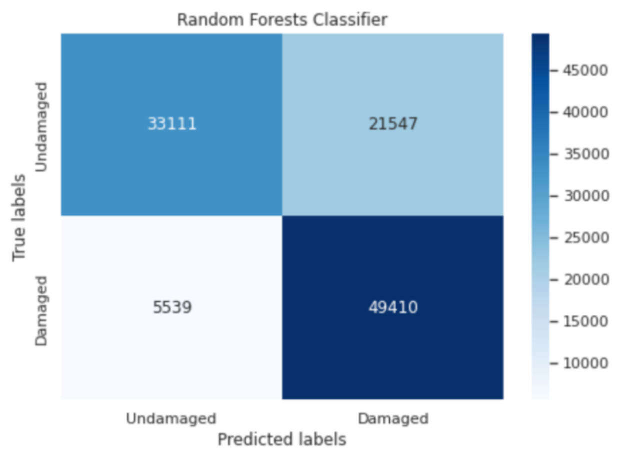


Overall training ratings of all of the ‘ existing to settle at around 0.8 (80%), suggesting that the machine's effectiveness will not change when we feed it new data. Changing the modeling is still an option; we'll do a comprehensive linear search for the best hyper parameters to use, restricting our self to 15,000 pieces of information in which the model's effectiveness hit a plateau, as shown in the graph above.

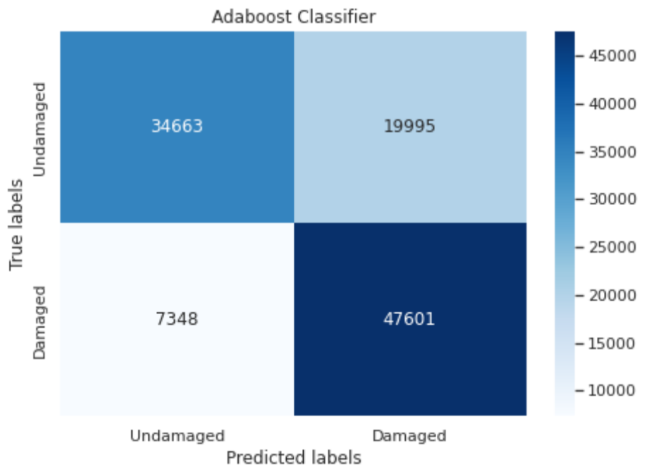
```
Best parameters for Gradient Boosting Classifier are: {'learning_rate': 0.1, 'max_depth': 3,
'n_estimators': 250}
Best parameters for Random Forests Classifier are: {'criterion': 'entropy', 'max_depth': 9,
'max_features': 'auto', 'n_estimators': 100}
Best parameters for Adaboost Classifier are: {'learning_rate': 1, 'n_estimators': 50}
```



Avg. Cross Validated Scores (Undamaged | Damaged)
F1-score: 73.38% | 78.25%
Recall : 66.00% | 86.12%
Precision: 82.62% | 71.70%



Avg. Cross Validated Scores (Undamaged | Damaged)
F1-score: 71.06% | 78.35%
Recall : 60.82% | 89.64%
Precision: 85.45% | 69.59%



Avg. Cross Validated Scores (Undamaged | Damaged)
F1-score: 71.94% | 77.53%
Recall : 63.98% | 86.11%
Precision: 82.17% | 70.51%

The finest three versions of the chosen models are shown top. Therefore, thankfully, these are superior in each and every aspect! But the issue remains: which would be the right choice?

The inhabitants' well-being is by far the most essential to me, hence why, notwithstanding the Adaboost classifier's significantly higher F1-score, We believe the random forest algorithm is the best overall.

.

V. Conclusion

There is indeed a compromise in all systems among categorizing more buildings as destroyed thus risk a higher accuracy but greater recall, or being more selective with categorization and sacrificing a lower recall but better precision. Random forest algorithm does have the found that the top of all 3 models, implying that it may incorrectly label certain structures as destroyed when they are not (lower precision).

The random forest classifier has a higher recall score for damaged buildings, and not a whole lot worse of a recall score for the undamaged ones compared to the other models, wasting some resources to unnecessarily check structurally sound buildings if this model was ever applied to a real life scenario, but preventing disaster and saving lives.

When contrasted to some other models, this random forest classifier has a greater recall score for structural failure and a similar recall score for intact buildings, wasting time and resources by checking good structural buildings needlessly, but avoiding tragedy and saving a life.

To summarize, the random forest model was train can: To summarize, the random forest model was trained can:

1) Identify approximately 90% of the structures that really are susceptible to destruction as ruined.

2) Mark roughly 60 percent of structures that aren't at risk of being harmed as "unharmed."

Consider the 70 percent of structures labeled as destroyed were indeed destroyed, whereas 85 percent of structures labeled as intact were genuinely undamaged.

VI. References

[1] X. Ding, X. Wang, A. Dou and '. Wang, "Study on a practical earthquake damage analysis and processing system based on RS and GIS," *2009 IEEE International Geoscience and Remote Sensing Symposium,* pp. IV-490-IV-493, 2009.

[2] K. A. Korkmaz and M. Abualkibash2, "Earthquake Damage Detection Using Before and After Earthquake Satellite Images," *2018 IEEE International Conference on Electro/Information Technology,* pp. 0615-0619, 2018.

[3] J. -W. Lin, C. -T. Chao and J. -S. Chiou, "Determining Neuronal Number in Each Hidden Layer Using Earthquake Catalogues as Training Data in Training an Embedded Back Propagation Neural Network for Predicting Earthquake Magnitude," *IEEE Access,* vol. 6, pp. 52582-52597, 2018.

[4] D. Dubois and R. Lepage, ""Automated building damage classification for the case of the 2010 Haiti earthquake."," *IEEE International Geoscience and Remote Sensing Symposium - IGARSS,* pp. 695-698, 2013.

[5] X. Wang, A. Dou, X. Ding and X. Yuan, ""The Development of Rapid Extraction and Publishing System of Earthquake Damage Based on Remote Sensing,"," *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium,* pp. 2921-2924, 2018.

[6] H. Gokon, S. Koshimura, J. Post, C. Geiß, E. Stein and M. Matsuoka, ""Detecting building damage caused by the 2011 Tohoku earthquake tsunami using TerraSAR-X data"," *2014 IEEE Geoscience and Remote Sensing Symposium,* pp. 1851-1854, 2014.

[7] W. Han, Y. Gan, S. Chen and X. Wang, ""Study on Earthquake Prediction Model Based on Traffic Disaster Data"," *2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS),* pp. 331-334, 2020.

[8] S. Huang, A. Dou, X. Wang and J. Wang, ""Earthquake-induced building damage detection method based on normal computation of neighboring points searching on 2D-plane"," *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS),* pp. 4251-4254, 2016.

[9] J. Yang and M. Xiang, ""The reclamation potential assessment of the damaged land by Wenchuan earthquake"," *2011 International Conference on Multimedia Technology,* pp. 1318-1321, 2011.

[10] C. Li and X. Liu, ""An improved PSO-BP neural network and its application to earthquake prediction"," *2016 Chinese Control and Decision Conference (CCDC),* pp. 3434-3438, 2016.

[11] C. P. Shabariram and K. E. Kannammal, ""Earthquake prediction using map reduce framework"," *International Conference on Computer Communication and Informatics (ICCCI),* pp. 1-6, 2017.

[12] D. Sun and B. Sun, ""Rapid prediction of earthquake damage to buildings based on fuzzy analysis"," *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery,* pp. 1332-1335, 2010.

[13] M. Syifa, S. Ryoo and C. -W. Lee, ""Post-Earthquake Damage Mapping Using Artificial Neural Network and Support Vector Machine Classifiers at Palu, Indonesia"," *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium* , pp. 9577-9580, 2019.

[14] X. Wei, X. Cui, C. Jiang and X. Zhou, ""The Earthquake Probability Prediction Based on Weighted Factor Coefficients of Principal Components"," *2009 Fifth International Conference on Natural Computation,* pp. 608-612, 2009.

[15] Z. Yan, R. Huazhong and G. Danyang, ""Research on the Detection Method of Building Seismic Damage Change"," *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium,* pp. 2906-2908, 2020.

[16] X. Ye, Q. Qin, J. Wang, J. Wang, X. Yang and X. Qin, ""Detecting damaged buildings caused by earthquake using local gradient orientation entropy statistics method,"," *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS),* pp. 3568-3571, 2015.