Big Data Project Report

Team Members

Deepesh (fxd180002@utdallas.edu)
Vamshidher Reddy Voncha (vxv170013@utdallas.edu)
Akshay Maganti (axm180074@utdallas.edu)

Introduction and Problem Description

The objective of this project is to help people find the best apartments in a locality/zip-code based on various factors like the amenities, nearby schools, user reviews on the property, expenses involved etc.

For example, when a user wants to move to a place near to his office and he wants to find a property with good amenities and school for his kids, then he can find the properties in nearby localities on the processed dataset and filter them on amenity and school score.

This project involves scoring the aforementioned entities by calculating the weights based on algorithms such as sentiment analysis, tf-idf and certain mathematical procedures.

The data required for this project has been web scraped from apartments.com. The code for the scraper is available at https://github.com/dc297/webscrapercs. The data is then stored into Sql Sever on Microsoft's Azure cloud.

Related Work

- Storing and accessing data from Sql Server on Azure Cloud.
- Setting up Spark Cluster on AWS.
- Setting Kafka cluster (certified by bitnami) on Google Cloud Platform to stream data into elastic search.
- Installing logstash on Kafka cluster and configuring it to push data into elastic search.
- Setting up Elasticsearch and Kibana to provide the final outputs.

DataSet Description

The dataset contains the following tables:

- AMENITY **[id, name,DESCRIPTION,AMENITY_type]**
- AMENITY_TYPE **[id, name]**
- PROPERTY_AMENITY_MAP **[id, propertyID,amenityID]**
- APARTMENT **[id, name, NO_OF_BEDS, NO_OF_BATH, minPrice, maxPrice, unit, area, name, availability, propertyID]**
- Expense_Type **[id, name]**
- Expenses **[id, name, cost(min-max),Expense_type, propertyID]**
- NearestTransitPoint_Category **[id, name]**
- NearestTransitPointInterest **[id, name, drive, distance, NTPCid]**
- NTPI_Property **[id, npti, propertyID]**
- PROPERTY **[id, name, address, zipcode, city, state, country, description, phone, email, propertyTypeID]**
- Property_school **[id, PropertyTypeID, SchoolID, PropertyID]**
- PROPERTY_TYPE **[id, proptyType]**
- review **[id, content, name, rating, propertyId]**
- School **[id, name, textType, grades, no_of_students, contact, rating, type]**
- URL

Preprocessing Techniques

Data Cleaning by removing the outliers, duplicates and redundant data and creating a database and its Tables as per our requirement.

Using the spark.read.jdbc() connector and passing custom SQL queries instead of table name to pull the required data from SQL server hosted on Azure into spark to process the scores for every property.

After the data has been processed, the calculated scores along with their respective property ID are pushed into kafka into their respective topics.

From kafka, the data is pushed into ElasticSearch using where we can visualize data using Kibana.

Proposed Solution and Methods

Scoring Methodologies

Amenities

The first step in producing the scores for amenities is to clean data to update the table PROPERTY_AMENITY_MAP to **[id, AmenityId, PropertyID].**

Now as we have the data in the required format, we calculate the weights of amenity. We use the same algorithm used to calculate tf-idf. We considered amenities as terms and properties as documents.

With this algorithm, we can give those amenities more weight which are available in fewer properties. Then we group by property and add its amenities weights to get its **AmenityScore**.

Expenses

We designed the algorithm to calculate the score for expenses, such that it is inversely proportional to the expenses and the rent they charge. The **ExpenseScore** can be calculated using the formula

maxMinValue/(min expense for that property) + maxMaxValue/(max expense for that property)

School

Calculating school score by just taking the average rating of all the schools near that property is not enough. So, we also add weightage to the strength of the school. We calculate the SchoolScore using the formula

(average strength of schools) / minAvgStudentsValue + (average rating of schools) / minAvgSchoolRatingValue

The data send by this class into kafka includes the information of the score, school with best rating and maximum number of students in that zipcode.

Review

The ReviewScore is a combination of average ratings (on a scale of 1 - 5) for the property and average of the score (ranging from 1 - 5) obtained by sentiment analysis of all the reviews we have for that property. To calculate the sentiment score, we used a pretrained model generated by Stanford core NLP.

Sentiment analysis score legend:

1 – Very Negative

2 – Negative

3 – Neutral

4 – Positive

5 – Very positive

Kafka Usage

The kafka cluster is hosted on google cloud platform. The following topics are used in the transfer of processed data:

- amenity
- schoolscore

- reviewscore
- expense
- address
- name
- zip
- schoolname
- schoolrating
- minprice
- maxprice

The producer created in the spark application is configured to push the data into their respective topics. As property_id is the key in all the cases, we define only one kind of key serializer and deserializer i.e., IntegerSerializer and Deserializer. But in the case of value, we send both double values and string values. We configure the producer according to our requirement.

Logstash which is installed on the same cluster is configured to listen all the topics. The exact configuration of logstash is available in the source code in the file logstash-simple.conf. The data is then sent to **elastic search**.

Elastic Search

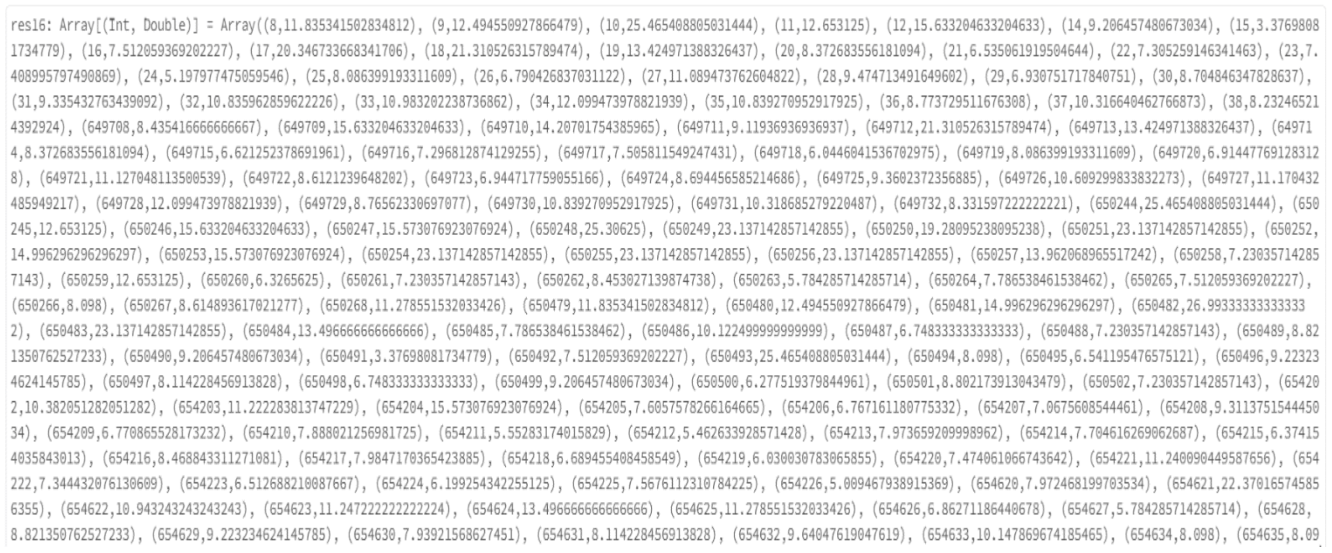
In Elasticsearch, the data is stored in index `property` with the following mappings:

```
address" : {
  "type" : "text"
},
"amenity" : {
  "type" : "float"
},
"expense" : {
  "type" : "float"
},
"maxprice" : {
  "type" : "float"
},
"minprice" : {
  "type" : "float"
},
"name" : {
```

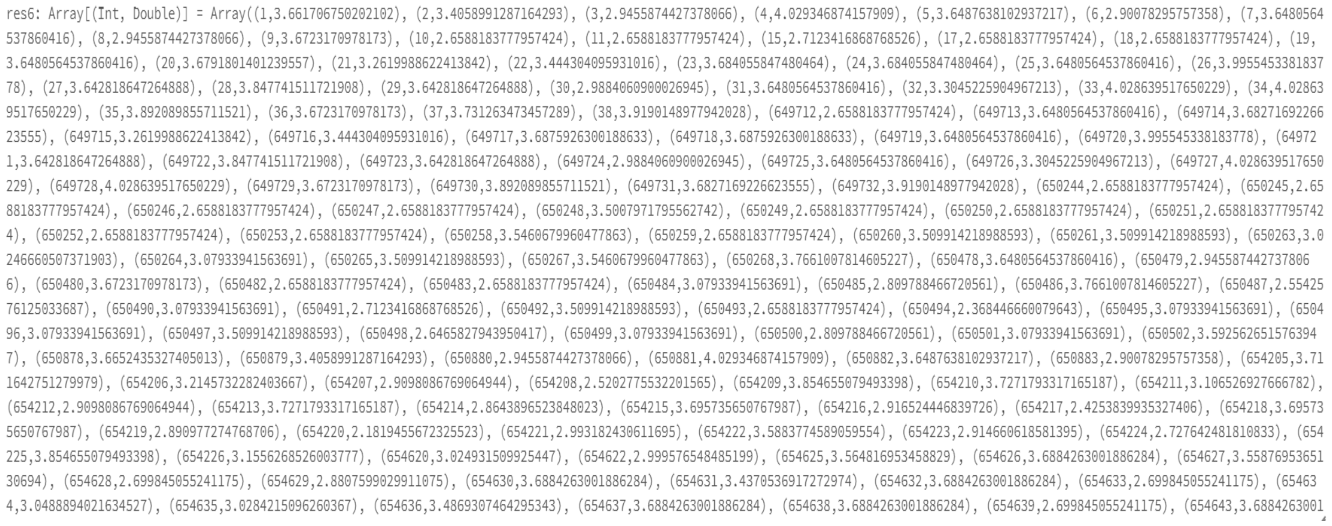
```
            "type" : "text"
          },
          "reviewscore" : {
            "type" : "float"
          },
          "schoolname" : {
            "type" : "text"
          },
          "schoolrating" : {
            "type" : "float"
          },
          "schoolscore" : {
            "type" : "float"
          },
         "zip" : {
            "type" : "text"
          }
```
This index is then used to create an index-pattern in Kibana and filter/search/sort the data.

Experimental results and Analysis:

Sample result of the property id and their corresponding weights for the Expense data

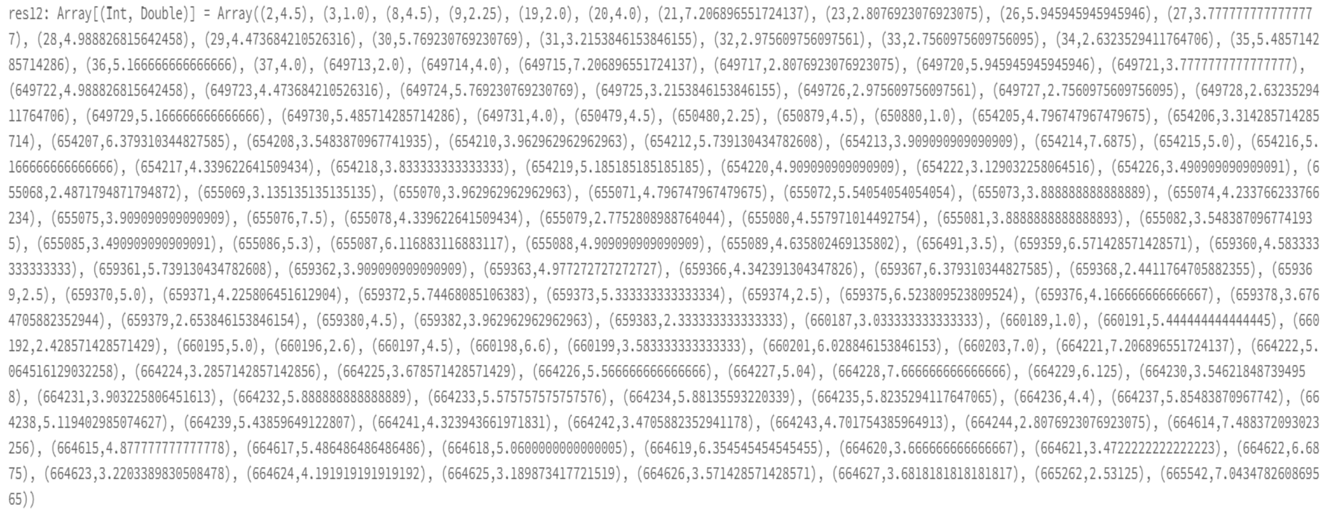

```
res16: Array[(Int, Double)] = Array((8,11.835341502834812), (9,12.494550927866479), (10,25.465408805031444), (11,12.653125), (12,15.633204633204633), (14,9.206457480673034), (15,3.3769808
1734779), (16,7.512059369202227), (17,20.346733668341706), (18,21.310526315789474), (19,13.424971388326437), (20,8.372683556181094), (21,6.535061919504644), (22,7.305259146341463), (23,7.
408995797490869), (24,5.197977475059546), (25,8.086399193311609), (26,6.790426837031122), (27,11.089473762604822), (28,9.474713491649602), (29,6.930751717840751), (30,8.704846347828637),
(31,9.335432763439092), (32,10.835962859622226), (33,10.983202238736862), (34,12.099473978821939), (35,10.839270952917925), (36,8.773729511676308), (37,10.316640462766873), (38,8.23246521
4392924), (649708,8.435416666666667), (649709,15.633204633204633), (649710,14.20701754385965), (649711,9.11936936936937), (649712,21.310526315789474), (649713,13.424971388326437), (64971
4,8.372683556181094), (649715,6.621252378691961), (649716,7.296812874129255), (649717,7.505811549247431), (649718,6.0446041536702975), (649719,8.086399193311609), (649720,6.91447769128312
8), (649721,11.127048113500539), (649722,8.6121239648202), (649723,6.944717759055166), (649724,8.694456585214686), (649725,9.3602372356885), (649726,10.609299833832273), (649727,11.170432
485949217), (649728,12.099473978821939), (649729,8.76562330697077), (649730,10.839270952917925), (649731,10.318685279220487), (649732,8.331597222222221), (650244,25.465408805031444), (650
245,12.653125), (650246,15.633204633204633), (650247,15.573076923076924), (650248,25.30625), (650249,23.137142857142855), (650250,19.28095238095238), (650251,23.137142857142855), (650252,
14.996296296296297), (650253,15.573076923076924), (650254,23.137142857142855), (650255,23.137142857142855), (650256,23.137142857142855), (650257,13.962068965517242), (650258,7.23035714285
7143), (650259,12.653125), (650260,6.3265625), (650261,7.230357142857143), (650262,8.453027139874738), (650263,5.784285714285714), (650264,7.786538461538462), (650265,7.512059369202227),
(650266,8.098), (650267,8.614893617021277), (650268,11.278551532033426), (650479,11.835341502834812), (650480,12.494550927866479), (650481,14.996296296296297), (650482,26.99333333333333
2), (650483,23.137142857142855), (650484,13.496666666666666), (650485,7.786538461538462), (650486,10.122499999999999), (650487,6.748333333333333), (650488,7.230357142857143), (650489,8.82
1350762527233), (650490,9.206457480673034), (650491,3.37698081734779), (650492,7.512059369202227), (650493,25.465408805031444), (650494,8.098), (650495,6.541195476575121), (650496,9.22323
4624145785), (650497,8.114228456913828), (650498,6.748333333333333), (650499,9.206457480673034), (650500,6.277519379844961), (650501,8.802173913043479), (650502,7.230357142857143), (65420
2,10.382051282051282), (654203,11.222283813747229), (654204,15.573076923076924), (654205,7.6057578266164665), (654206,6.767161180775332), (654207,7.0675608544461), (654208,9.3113751544450
34), (654209,6.770865528173232), (654210,7.888021256981725), (654211,5.55283174015829), (654212,5.462633928571428), (654213,7.973659209998962), (654214,7.704616269062687), (654215,6.37415
4035843013), (654216,8.468843311271081), (654217,7.9847170365423885), (654218,6.689455408458549), (654219,6.030030783065855), (654220,7.474061066743642), (654221,11.240090449587656), (654
222,7.344432076130609), (654223,6.512688210087667), (654224,6.199254342255125), (654225,7.5676112310784225), (654226,5.009467938915369), (654620,7.972468199703534), (654621,22.37016574585
6355), (654622,10.943243243243243), (654623,11.247222222222224), (654624,13.496666666666666), (654625,11.278551532033426), (654626,6.86271186440678), (654627,5.784285714285714), (654628,
8.821350762527233), (654629,9.223234624145785), (654630,7.93921568627451), (654631,8.114228456913828), (654632,9.64047619047619), (654633,10.147869674185465), (654634,8.098), (654635,8.09
```

Sample result of the property id and their corresponding weights for the School data



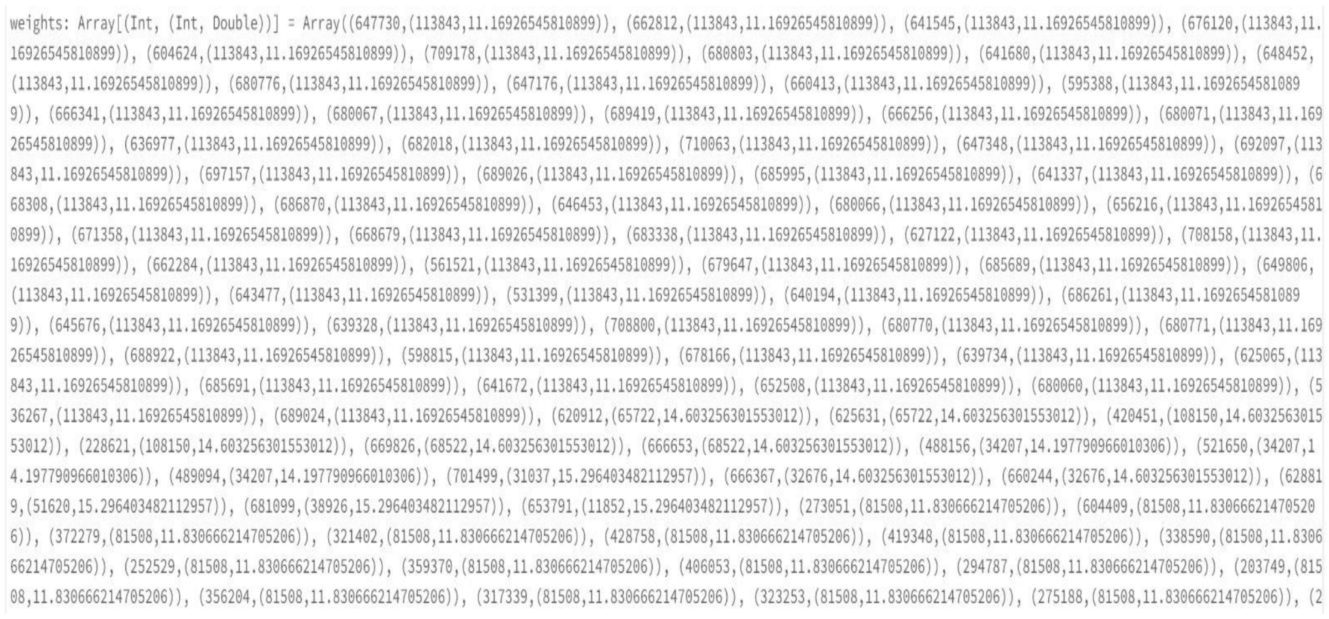
```
res6: Array[(Int, Double)] = Array((1,3.661706750202102), (2,3.4058991287164293), (3,2.9455874427378066), (4,4.029346874157909), (5,3.6487638102937217), (6,2.90078295757358), (7,3.6480564
537860416), (8,2.9455874427378066), (9,3.6723170978173), (10,2.6588183777957424), (11,2.6588183777957424), (15,2.7123416868768526), (17,2.6588183777957424), (18,2.6588183777957424), (19,
3.6480564537860416), (20,3.6791801401239557), (21,3.2619988622413842), (22,3.444304095931016), (23,3.684055847480464), (24,3.684055847480464), (25,3.6480564537860416), (26,3.9955453381837
78), (27,3.642818647264888), (28,3.847741511721908), (29,3.642818647264888), (30,2.9884060900026945), (31,3.6480564537860416), (32,3.3045225904967213), (33,4.028639517650229), (34,4.02863
9517650229), (35,3.892089855711521), (36,3.6723170978173), (37,3.731263473457289), (38,3.9190148977942028), (649712,2.6588183777957424), (649713,3.6480564537860416), (649714,3.68271692266
23555), (649715,3.2619988622413842), (649716,3.444304095931016), (649717,3.6875926300188633), (649718,3.6875926300188633), (649719,3.6480564537860416), (649720,3.995545338183778), (64972
1,3.642818647264888), (649722,3.847741511721908), (649723,3.642818647264888), (649724,2.9884060900026945), (649725,3.6480564537860416), (649726,3.3045225904967213), (649727,4.028639517650
229), (649728,4.028639517650229), (649729,3.6723170978173), (649730,3.892089855711521), (649731,3.6827169226623555), (649732,3.9190148977942028), (650244,2.6588183777957424), (650245,2.65
88183777957424), (650246,2.6588183777957424), (650247,2.6588183777957424), (650248,3.5007971795562742), (650249,2.6588183777957424), (650250,2.6588183777957424), (650251,2.658818377795742
4), (650252,2.6588183777957424), (650253,2.6588183777957424), (650258,3.5460679960477863), (650259,2.6588183777957424), (650260,3.509914218988593), (650261,3.509914218988593), (650263,3.0
246660507371903), (650264,3.07933941563691), (650265,3.509914218988593), (650267,3.5460679960477863), (650268,3.7661007814605227), (650478,3.6480564537860416), (650479,2.945587442737806
6), (650480,3.6723170978173), (650482,2.6588183777957424), (650483,2.6588183777957424), (650484,3.07933941563691), (650485,2.809788466720561), (650486,3.7661007814605227), (650487,2.55425
76125033687), (650490,3.07933941563691), (650491,2.7123416868768526), (650492,3.509914218988593), (650493,2.6588183777957424), (650494,2.368446660079643), (650495,3.07933941563691), (6504
96,3.07933941563691), (650497,3.509914218988593), (650498,2.6465827943950417), (650499,3.07933941563691), (650500,2.809788466720561), (650501,3.07933941563691), (650502,3.592562651576394
7), (650878,3.6652435327405013), (650879,3.4058991287164293), (650880,2.9455874427378066), (650881,4.029346874157909), (650882,3.6487638102937217), (650883,2.90078295757358), (654205,3.71
1642751279979), (654206,3.2145732282403667), (654207,2.9098086769064944), (654208,2.5202775532201565), (654209,3.854655079493398), (654210,3.7271793317165187), (654211,3.106526927666782),
(654212,2.9098086769064944), (654213,3.7271793317165187), (654214,2.8643896523848023), (654215,3.695735650767987), (654216,2.916524446839726), (654217,2.4253839935327406), (654218,3.69573
5650767987), (654219,2.890977274768706), (654220,2.1819455672325523), (654221,2.993182430611695), (654222,3.5883774589059554), (654223,2.914660618581395), (654224,2.727642481810833), (654
225,3.854655079493398), (654226,3.1556268526003777), (654620,3.024931509925447), (654622,2.999576548485199), (654625,3.564816953458829), (654626,3.6884263001886284), (654627,3.55876953651
30694), (654628,2.699845055241175), (654629,2.8807599029911075), (654630,3.6884263001886284), (654631,3.4370536917272974), (654632,3.6884263001886284), (654633,2.699845055241175), (65463
4,3.0488894021634527), (654635,3.0284215096260367), (654636,3.4869307464295343), (654637,3.6884263001886284), (654638,3.6884263001886284), (654639,2.699845055241175), (654643,3.6884263001
```

Sample result of the property id and their corresponding weights for the Review data



```
res12: Array[(Int, Double)] = Array((2,4.5), (3,1.0), (8,4.5), (9,2.25), (19,2.0), (20,4.0), (21,7.206896551724137), (23,2.8076923076923075), (26,5.945945945945946), (27,3.777777777777777
7), (28,4.988826815642458), (29,4.473684210526316), (30,5.769230769230769), (31,3.2153846153846155), (32,2.975609756097561), (33,2.7560975609756095), (34,2.6323529411764706), (35,5.4857142
85714286), (36,5.166666666666666), (37,4.0), (649713,2.0), (649714,4.0), (649715,7.206896551724137), (649717,2.8076923076923075), (649720,5.945945945945946), (649721,3.7777777777777777),
(649722,4.988826815642458), (649723,4.473684210526316), (649724,5.769230769230769), (649725,3.2153846153846155), (649726,2.975609756097561), (649727,2.7560975609756095), (649728,2.63235294
11764706), (649729,5.166666666666666), (649730,5.485714285714286), (649731,4.0), (650479,4.5), (650480,2.25), (650879,4.5), (650880,1.0), (654205,4.796747967479675), (654206,3.314285714285
714), (654207,6.379310344827585), (654208,3.5483870967741935), (654210,3.962962962962963), (654212,5.739130434782608), (654213,3.909090909090909), (654214,7.6875), (654215,5.0), (654216,5.
166666666666666), (654217,4.339622641509434), (654218,3.8333333333333333), (654219,5.185185185185185), (654220,4.909090909090909), (654222,3.129032258064516), (654226,3.490909090909091), (6
55068,2.4871794871794872), (655069,3.135135135135135), (655070,3.962962962962963), (655071,4.796747967479675), (655072,5.54054054054054), (655073,3.888888888888889), (655074,4.233766233766
234), (655075,3.909090909090909), (655076,7.5), (655078,4.339622641509434), (655079,2.7752808988764044), (655080,4.557971014492754), (655081,3.8888888888888893), (655082,3.548387096774193
5), (655085,3.490909090909091), (655086,5.3), (655087,6.116883116883117), (655088,4.909090909090909), (655089,4.635802469135802), (656491,3.5), (659359,6.571428571428571), (659360,4.583333
333333333), (659361,5.739130434782608), (659362,3.909090909090909), (659363,4.977272727272727), (659366,4.342391304347826), (659367,6.379310344827585), (659368,2.4411764705882355), (65936
9,2.5), (659370,5.0), (659371,4.225806451612904), (659372,5.74468085106383), (659373,5.333333333333334), (659374,2.5), (659375,6.523809523809524), (659376,4.166666666666667), (659378,3.676
4705882352944), (659379,2.6538461538461538), (659380,4.5), (659382,3.962962962962963), (659383,2.3333333333333333), (660187,3.0333333333333333), (660189,1.0), (660191,5.444444444444445), (660
192,2.428571428571429), (660195,5.0), (660196,2.6), (660197,4.5), (660198,6.6), (660199,3.5833333333333333), (660201,6.028846153846153), (660203,7.0), (664221,7.206896551724137), (664222,5.
064516129032258), (664224,3.2857142857142856), (664225,3.678571428571429), (664226,5.566666666666666), (664227,5.04), (664228,7.666666666666666), (664229,6.125), (664230,3.54621848739495
8), (664231,3.903225806451613), (664232,5.888888888888889), (664233,5.575757575757576), (664234,5.88135593220339), (664235,5.8235294117647065), (664236,4.4), (664237,5.85483870967742), (66
4238,5.119402985074627), (664239,5.43859649122807), (664241,4.323943661971831), (664242,3.4705882352941178), (664243,4.701754385964913), (664244,2.8076923076923075), (664614,7.488372093023
256), (664615,4.877777777777778), (664617,5.486486486486486), (664618,5.0600000000000005), (664619,6.354545454545455), (664620,3.666666666666667), (664621,3.4722222222222223), (664622,6.68
75), (664623,3.2203389830508478), (664624,4.191919191919192), (664625,3.189873417721519), (664626,3.571428571428571), (664627,3.6818181818181817), (665262,2.53125), (665542,7.0434782608695
65))
```

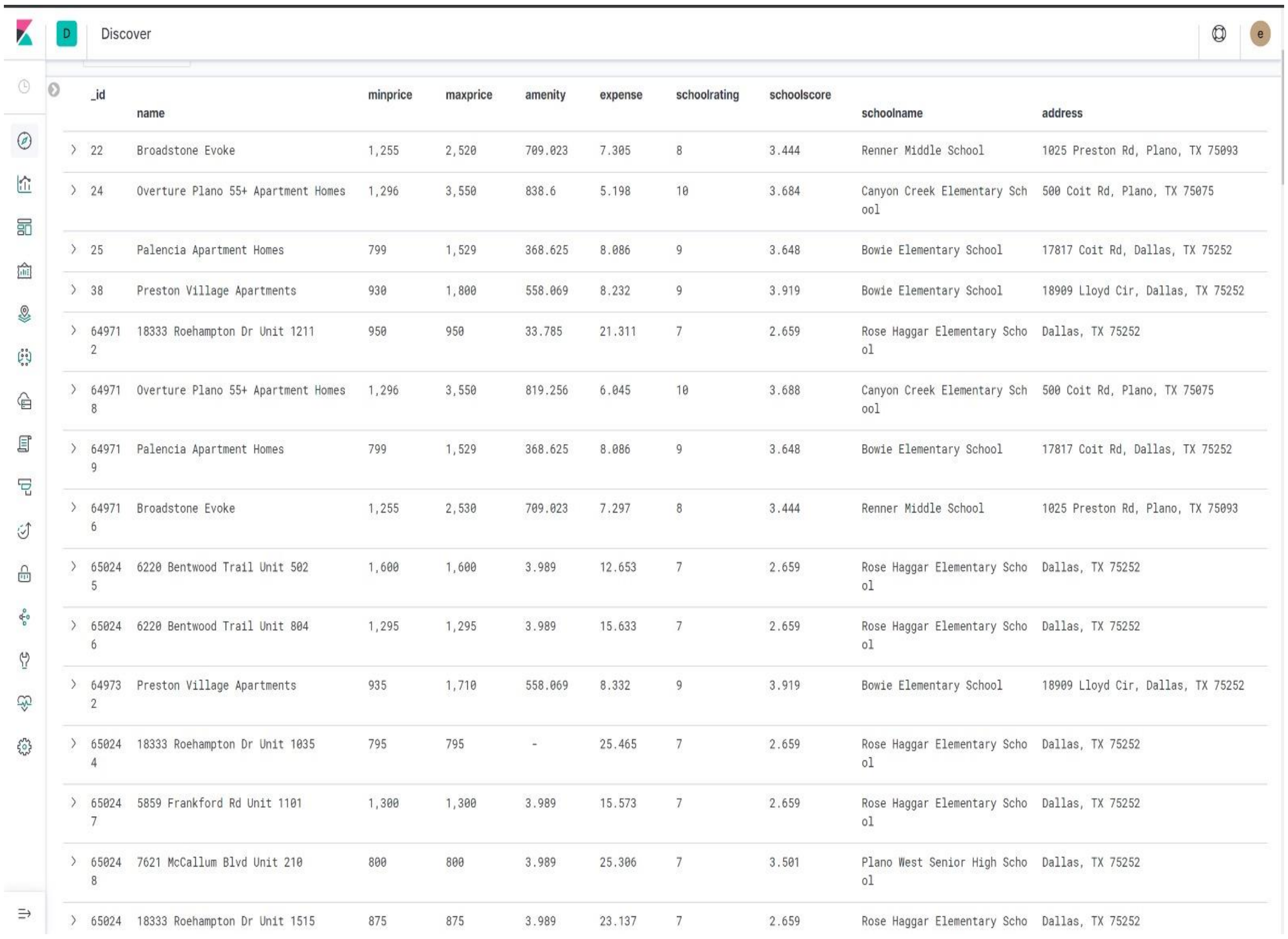
Sample result of the property id and their corresponding weights for the Amenities data



```
weights: Array[(Int, (Int, Double))] = Array((647730,(113843,11.16926545810899)), (662812,(113843,11.16926545810899)), (641545,(113843,11.16926545810899)), (676120,(113843,11.
16926545810899)), (604624,(113843,11.16926545810899)), (709178,(113843,11.16926545810899)), (680803,(113843,11.16926545810899)), (641680,(113843,11.16926545810899)), (648452,
(113843,11.16926545810899)), (680776,(113843,11.16926545810899)), (647176,(113843,11.16926545810899)), (660413,(113843,11.16926545810899)), (595388,(113843,11.1692654581089
9)), (666341,(113843,11.16926545810899)), (680067,(113843,11.16926545810899)), (689419,(113843,11.16926545810899)), (666256,(113843,11.16926545810899)), (680071,(113843,11.169
26545810899)), (636977,(113843,11.16926545810899)), (682018,(113843,11.16926545810899)), (710063,(113843,11.16926545810899)), (647348,(113843,11.16926545810899)), (692097,(113
843,11.16926545810899)), (697157,(113843,11.16926545810899)), (689026,(113843,11.16926545810899)), (685995,(113843,11.16926545810899)), (641337,(113843,11.16926545810899)), (6
68308,(113843,11.16926545810899)), (686870,(113843,11.16926545810899)), (646453,(113843,11.16926545810899)), (680066,(113843,11.16926545810899)), (656216,(113843,11.1692654581
0899)), (671358,(113843,11.16926545810899)), (668679,(113843,11.16926545810899)), (683338,(113843,11.16926545810899)), (627122,(113843,11.16926545810899)), (708158,(113843,11.
16926545810899)), (662284,(113843,11.16926545810899)), (561521,(113843,11.16926545810899)), (679647,(113843,11.16926545810899)), (685689,(113843,11.16926545810899)), (649806,
(113843,11.16926545810899)), (643477,(113843,11.16926545810899)), (531399,(113843,11.16926545810899)), (640194,(113843,11.16926545810899)), (686261,(113843,11.1692654581089
9)), (645676,(113843,11.16926545810899)), (639328,(113843,11.16926545810899)), (708800,(113843,11.16926545810899)), (680770,(113843,11.16926545810899)), (680771,(113843,11.169
26545810899)), (688922,(113843,11.16926545810899)), (598815,(113843,11.16926545810899)), (678166,(113843,11.16926545810899)), (639734,(113843,11.16926545810899)), (625065,(113
843,11.16926545810899)), (685691,(113843,11.16926545810899)), (641672,(113843,11.16926545810899)), (652508,(113843,11.16926545810899)), (680060,(113843,11.16926545810899)), (5
36267,(113843,11.16926545810899)), (689024,(113843,11.16926545810899)), (620912,(65722,14.603256301553012)), (625631,(65722,14.603256301553012)), (420451,(108150,14.6032563015
53012)), (228621,(108150,14.603256301553012)), (669826,(68522,14.603256301553012)), (666653,(68522,14.603256301553012)), (488156,(34207,14.197790966010306)), (521650,(34207,1
4.197790966010306)), (489094,(34207,14.197790966010306)), (701499,(31037,15.296403482112957)), (666367,(32676,14.603256301553012)), (660244,(32676,14.603256301553012)), (62881
9,(51620,15.296403482112957)), (681099,(38926,15.296403482112957)), (653791,(11852,15.296403482112957)), (273051,(81508,11.830666214705206)), (604409,(81508,11.83066621470520
6)), (372279,(81508,11.830666214705206)), (321402,(81508,11.830666214705206)), (428758,(81508,11.830666214705206)), (419348,(81508,11.830666214705206)), (338590,(81508,11.8306
66214705206)), (252529,(81508,11.830666214705206)), (359370,(81508,11.830666214705206)), (406053,(81508,11.830666214705206)), (294787,(81508,11.830666214705206)), (203749,(815
08,11.830666214705206)), (356204,(81508,11.830666214705206)), (317339,(81508,11.830666214705206)), (323253,(81508,11.830666214705206)), (275188,(81508,11.830666214705206)), (2
```

End Result

Result as Displayed on Kibana over Elastic Search



_id	name	minprice	maxprice	amenity	expense	schoolrating	schoolscore	schoolname	address
22	Broadstone Evoke	1,255	2,520	709.023	7.305	8	3.444	Renner Middle School	1025 Preston Rd, Plano, TX 75093
24	Overture Plano 55+ Apartment Homes	1,296	3,550	838.6	5.198	10	3.684	Canyon Creek Elementary School	500 Coit Rd, Plano, TX 75075
25	Palencia Apartment Homes	799	1,529	368.625	8.086	9	3.648	Bowie Elementary School	17817 Coit Rd, Dallas, TX 75252
38	Preston Village Apartments	930	1,800	558.069	8.232	9	3.919	Bowie Elementary School	18909 Lloyd Cir, Dallas, TX 75252
649712	18333 Roehampton Dr Unit 1211	950	950	33.785	21.311	7	2.659	Rose Haggar Elementary School	Dallas, TX 75252
649718	Overture Plano 55+ Apartment Homes	1,296	3,550	819.256	6.045	10	3.688	Canyon Creek Elementary School	500 Coit Rd, Plano, TX 75075
649719	Palencia Apartment Homes	799	1,529	368.625	8.086	9	3.648	Bowie Elementary School	17817 Coit Rd, Dallas, TX 75252
649716	Broadstone Evoke	1,255	2,530	709.023	7.297	8	3.444	Renner Middle School	1025 Preston Rd, Plano, TX 75093
650245	6220 Bentwood Trail Unit 502	1,600	1,600	3.989	12.653	7	2.659	Rose Haggar Elementary School	Dallas, TX 75252
650246	6220 Bentwood Trail Unit 804	1,295	1,295	3.989	15.633	7	2.659	Rose Haggar Elementary School	Dallas, TX 75252
649732	Preston Village Apartments	935	1,710	558.069	8.332	9	3.919	Bowie Elementary School	18909 Lloyd Cir, Dallas, TX 75252
650244	18333 Roehampton Dr Unit 1035	795	795	-	25.465	7	2.659	Rose Haggar Elementary School	Dallas, TX 75252
650247	5859 Frankford Rd Unit 1101	1,300	1,300	3.989	15.573	7	2.659	Rose Haggar Elementary School	Dallas, TX 75252
650248	7621 McCallum Blvd Unit 210	800	800	3.989	25.306	7	3.501	Plano West Senior High School	Dallas, TX 75252
65024	18333 Roehampton Dr Unit 1515	875	875	3.989	23.137	7	2.659	Rose Haggar Elementary Scho	Dallas, TX 75252

Conclusion

We have successfully shown that there is a possibility to automatically rank apartments or properties located in multiple zip-codes. We modeled the problem as a ranking system.

We applied Sentiment analysis, tf-idf methodologies and used multiple libraries such as the Stanford NLP library for text mining. Due to the complexity of the problem at hand, we believe there is still a lot of work to overcome skewness.

The textual features that we used were based on some latest text mining ones that considered the sentiment from the reviews and the relevance among the data.

More advanced NLP techniques and the application usage monitoring can be added to perform better analysis as per the user requirement.

Feature selection methodologies such as the Principal Component Analysis can be used as the feature space is a high dimensional dataset with high intercorrelation. Simple techniques such as sampling of the data can be employed.

Upon examining the results, we find them to be as expected, the properties with higher amenity score had better and rarer amenities. Similarly, the results for school, expense and review also make sense.

Contribution of Team Members

1. Deepesh
    a. Setting up the environment for SQL server over Azure Cloud.
    b. Working with the elimination of duplicate data on SQL server.
    c. Configuring the Kafka over GCP, Elastic Search and Logstash.
    d. Initial formulation and consideration of the features.

2. Akshay Maganti
    a. Build and refactoring the Scala code on IntelliJ.
    b. Score calculation for the Amenities using tf-idf techniques.
    c. Monitoring and Debugging the functionality on AWS.
    d. Took care of Dependency Injection in the spark application.
    e. Initial formulation and consideration of the features.

3. Vamshider Reddy Voncha
    a. Query creation and optimization on SQL server.
    b. Scoring and analysis of the initial data related to expenses, schools, reviews using sentiment analysis and ratings of the properties.
    c. Creation of mathematical formulae related to the calculation of the weights for the entities available.

d. Analysis of the results and the correlation between the entities.

References

Data - https://www.apartments.com/

Libraries - https://stanfordnlp.github.io/CoreNLP/

Kafka - https://kafka.apache.org/documentation/

Elastic Search - https://www.elastic.co/guide/index.html

Kibana - https://www.elastic.co/guide/en/kibana/current/index.html

Custom Debugging and Error Handling - https://stackoverflow.com/