CSE 564 Project 2 Report

YouTube Link - https://youtu.be/_oCvVDlmIFY

Name: Aditya Ravindra Gaikwad
SBU ID: 112686420

1. Dataset description

- The data set used is called the “Ames Housing Dataset”. It describes the sale of individual residential property in Ames, Iowa from 2006 to 2010. The data set contains 2930 observations and a large number of explanatory variables (23 nominal, 23 ordinal, 14 discrete, and 20 continuous) involved in assessing home values.
- For this project, I have chosen 15 variables - SalePrice, GrLivArea, OverallQual, SaleCondition, BldgType, YearBuilt, GarageCars, LotArea, HeatingQC, KitchenQual, MiscVal, YearRemodAdd, YrSold, RoofStyle, SaleType.
- The Pandas library in Python was used to filter out/drop any “NULL” values in the dataset which resulted in a reduction of observations to 1460 rows**.**
- The preprocessing code is available in the Jupyter Notebook labeled “data_cleaning_and_testing.ipynb”
- The categorical variables were then converted to their numerical equivalent using a custom encoding so as to preserve the meaning of the category.

  Sample code:

```
# HeatingQC
encode_dict = {
    "Ex" : 5,
    "Gd" : 4,
    "TA" : 3,
    "Fa" : 2,
    "Po" : 1
}

data_df["HeatingQC"] = data_df.HeatingQC.map(encode_dict)
```

- The final filtered data frame was again stored as a new CSV to be used in the project.
- The other datasets for housing prices like the Boston Housing Dataset have very fewer observations as compared to this dataset which makes it a very good candidate for finding trends in housing prices.

2. Functionalities implemented

2.1. Efficient data storing

- The flask code and APIs are written so as to minimize re-execution of code and hence reduce redundancy.
- Once the samples are generated, they are stored in memory as long as the Flask app is running.
- When the user wants to see any of the graphs again, the APIs simply check if the same data exists in the RAM already and if it does, it simply returns that data.
- This saves us a lot of computation time and power.
- The data is stored as global variables in the form of dictionaries which is one of the most versatile data structures in Python.
- Code snippet -

```
records = {}

records["whole_dataset"] = []
records["stratified_sample"] = []
records["random_sample"] = []

records["data_loaded"] = False
records["cumulative_pca_variance_ratio"] = {}
records["intrinsic_dimensionality"] = {}

saved_pca = {}

task3 = {}
task3["top_2_pca"] = {}
task3["top_3_loaded_attributes"] = {}
task3["dataset"] = {}
task3["mds"] = {}
```

2.2. Accordion dropdown

- Bootstrap's accordion format was used to provide the user with a more comfortable way to navigate through the sections.
- The user can focus on the current graphs without being distracted by the other graphs.
- The template looks like:

```
<div id="accordion">
    <div class="card" id="task12">
        <div class="card-header">
            <a class="collapsed card-link" data-toggle="collapse"
href="#collapseOne">
                <i class="fa fa-angle-down rotate-icon"></i>  Task 1 & 2:
Generate samples and plot Scree Plots
            </a>
        </div>
        <div id="collapseOne" class="collapse" data-parent="#accordion">
            <div class="card-body">
            </div>
        </div>
```

```
    </div>
</div>
```

2.3. Scree Plot

- The Scree Plot for the whole dataset is shown by default.
- Users can choose which sample's Scree Plot they want to see.
- The API endpoint "/api/generate_data" is used to generate data needed by the Scree Plots.

The code snippet for the Scree Plot -

```
g.append("g")
    .selectAll(".bar")
    .data(data)
    .enter().append("rect")
    .attr("class", "bar")
    .attr("x", function(d) { return xAxis("feature "+ String(d.feature));})
    .attr("y", function(d) { return yAxis1(d.pca_variance);})
    .attr("width", xAxis.bandwidth())
    .attr("height", function(d) { return height - yAxis1(d.pca_variance); });

var curve = d3.line()
        .x(function(d) { return xAxis("feature "+ String(d.feature)); })
        .y(function(d) { return yAxis2(d.pca_variance_ratio); })
        .curve(d3.curveMonotoneX);

g.append("g")
    .append("path")
    .datum(cum_data) // Binds data to the line
    .attr("class", "line")
    .attr("d", curve);
```

2.4. Scatterplot

- The scatterplots for all 3 datasets are shown together so that the user can easily compare them and get a better insight.
- The API endpoint "/api/scatterplot_data" is used to generate the data needed for the scatter plots as well as the scatterplot matrix.
- The API is hit using an asynchronous AJAX POST call and data is transferred in JSON format.
- This ensures that the user gets a smooth experience.
- The 3 attributes with the highest PCA loadings are calculated in the function "get_loaded_attributes()" in app.py
- There are onclick events listening for opening of the task 3 accordion as well as the individual tabs inside the accordion.

Code snippet -

```
$('#task31').on('show.bs.collapse', function () {
    //on clicking the accordion menu for task 31
    scatterPlotMatrix = false;
    call_task3_api();
```

```
});

       document.getElementById("t3_top_2_pca").addEventListener("click", function()
{
     plotScatterPlot("whole_dataset", false, null);
     plotScatterPlot("random_sample", false, null);
     plotScatterPlot("stratified_sample", false, null);
});
```

The code snippet for the scatterplot -

```
g.selectAll(".dot")
 .data(data)
 .enter().append("circle")
 .attr("class", "dot")
 .attr("r", 3.5)
 .attr("cx", function(d) { return x(d.MDS_1); })
 .attr("cy", function(d) { return y(d.MDS_2); });
```

2.5. Scatterplot Matrix

- The data for the scatterplot matrix is loaded along with the data for scatterplots.
- The scatterplot matrix has a nice feature where the user can select an area from one scatterplot of two variables and then see the corresponding data elements highlighted in all the other scatterplots.
- This helps the user get a good idea about the data and how it's correlated with the other variables.

  The code snippet for the scatterplot matrix -

```
cell.append("rect")
    .attr("class", "frame")
    .attr("x", padding / 2)
    .attr("y", padding / 2)
    .attr("width", size - padding)
    .attr("height", size - padding);

cell.selectAll("circle")
    .data(data)
.enter().append("circle")
     .attr("cx", function(d) { return x(d[p.x]); })
    .attr("cy", function(d) { return y(d[p.y]); })
    .attr("r", 4)
    .style("fill", "#69b3a2");
```

3. Observations of data and visualization

3.1. Observations based on Scree plots

- The overall trend of explained variance between the three samples is almost the same.
- However, the stratified sample has the highest variance been captured by a smaller number of PCA features.

3.2. Observations from the Scatterplots

- The top 2 PCA scatterplots clearly suggest that the top 2 PCA vectors have an almost linear relationship.
- The data is scattered near a line in all 3 datasets.
- Even in the case of MDS with euclidean distance, the relationship of the vectors is still a linear one. However, in the random and stratified samples, the data is skewed towards Vector 1, suggesting that it has more pull or weightage on the data.
- In the case of MDS with correlation distance, the data is scattered a bit farther away and even here Vector 1 has more weightage.

3.3. Observations from the Scatterplot Matrices

- The top 3 variables with the highest PCA loadings in all 3 datasets are Sale Condition, Building Type and Year Sold.
- It is interesting to note that all 3 of these are categorical variables that were custom encoded.
- When we consider the full dataset, these variables are fairly randomly scattered in all the plots. They do not seem to have any relation between them.
- One interesting observation was that the number of entries in the dataset for sale conditions 1, 6 were almost equal to 50% of the total entries.
- Also, BuildingType 1 accounts for a lot of the records present.

3.4. Comparing the visualization alternatives used

- Using Scree plots to find the intrinsic dimensionality of the dataset is a good approach because it helps us find the minimum number of useful factors needed which account for more variance than could be expected by chance (75%).
- Scatterplots, on the other hand, are more useful in determining relations between two variables as well as see the spread of the variables and the major areas of density.
- Scatterplot matrices also have similar usage but offer a more compressed visualization of more variables which can be more convenient in some cases.

3.5. Effects of downsampling as compared to original data

- When looking at the Scatterplot matrices, data present in the random sample, when compared with the whole dataset, looks fairly random.
- However, in the case of **stratified sampling**, all the records having the **SaleCondition** as **3 & 4** have been totally **ignored**.
- As stratified sampling gets us a sample with records about the greater population of interest, this suggests that the records having SaleCondition as 3 & 4 were very low comparatively.
- The top 2 PCA representation Scatterplots of the whole dataset and stratified sample are very much alike, just that stratified sample sticks to the majority data. The scatterplot of the random sample has the exact opposite slope of its line.

References:

1. De Cock, Dean. "Ames, Iowa: Alternative to the Boston housing data as an end of semester regression project." Journal of Statistics Education 19.3 (2011).
2. D3V4 Scatterplot
3. Scatterplot Matrix Brushing d3v4