**Code Analysis Report**

**Number lines of Code**: 195

**Languages**: Py Spark and R.

**Software**: Zeppelin

**Work Flow for the Code**

Data loading, Cleaning of Data and Combining of Data.

Identifying the language interpreter that reduces the bottleneck of the code

Implementation of code.

**Code Performance Analysis**

There were two bottle necks we encountered in our code as defined in the table below. The place we experience the first roadblock is when we tried to bind the 2 data sets. This is where we used the R interpreter which helped us made implementation faster. The second roadblock was when we ran the linear regression it took us close to 5 min to run the linear regression on all the variables . Hence we tried to make the implementation faster we split the number regression steps to make implementation faster.

**Tables Describes the Performance of the Code**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Description** | **Duration** | **Language** | **Bottleneck** | **Comments** |
| Loading of Data | 3 min | PySpark | No Bottlenecks |  |
| Cleaning of Data | 2 mins | PySpark | No Bottlenecks |  |
| Binding of Data | 5 mins | PySpark | too long to bind the data the system stuck | We switched to the R interpret to reduce the binding time. R interpreter took 2 min to run the code |
| Creating the RDD pairs | 2 mins | PySpark | None |  |
| Running the linear Regression | 5 mins | R | Yes, its too long for the linear regression to run | We split the variable and ran the regression |
| Ploting the graphs | 3 mins | R | No |  |