7CS512 Business Analytics

Ntoulmperis Michail[1]
[1]107 Patision anenue & Pellinis Str, 11251, Athens, GR
UoD: 100615926, mich.ntoulmperis@mc-class.gr
MSc Big Data Analytics

Abstract. This study reviews a dataset of basketball players through Seasons 1999-2020 and from 49 Leagues to create a story telling data analysis report. By preprocessing the original data set, either by renaming some of the variables or even removing observations with missing values, we try to bring the data set in the proper form so we can use canonical correlation to select the proper regressor variables. Finally, a linear model was built and evaluated, a MIS Menu System was created along with a UML Diagram-Code Flow Chart.

Keywords: Dataset, MIS menu system, Data analysis, Pre-processing, Missing Values, Canonical Correlation, Linear Regression

Contents

1 Introduction

In the last decade, it has become common knowledge that how we process, analyze and visualize Big Data is going to affect the way the world works. Big data refers to data sets that are so large that conventional database management and data analysis tools are unable to work with them. While there isn't a scientific term that we can use in order to define exactly what Big data is, the most common ways describing it is with use of the ''5Vs''.[1] The ''5Vs'' of Big Data refer to: the high Volume, the high velocity, the variety, the high value and the low veracity.

The high volume indicates the huge size some data sets can be, with some of them exceeding multiple terabytes or even petabytes[2]. High velocity refers to the amazing speed that data is generated. while variety refers two complexities of big data. For example, data sets from a health clinic can be expected to have a vast number of combinations of numerical measurements, names of patients or even images and videos from different diagnosis. Furthermore, the high value of big data can be obvious, as it can reveal great insight to what exactly the data was by giving it meanings and what it can be. Lasty, low veracity corresponds to the changed uncertainty and the large-scale missing values of big data. Along with the growing size of datasets, data itself can often change or be corrupted.

In this paper we tried to demonstrate those aspects of Big Data by picking for our study a data set with a large number of observations and large variety of variables. We hope that through the proper pre-processing we can prepare the data in a way that is ready to be analyzed. By this we mean, that special care must be given to removal of missing values.

Furthermore, our main objective is to try and estimate a variable through analyzing the data. This objective can be achieved through studying the canonical correlation of each the value we want to predict with the rest of the variables, so we can optimize our model.

The method of supervised machine learning that we are going to use is Linear Regression. By splitting the data set we are going to try, and train evaluate a model. If this model proves to be fitting in our we will try and train it. Lastly statistical measures and metrics our going to be given in order to gain more insight of our data, while understanding which variable is the most statistically significant to the variable we want to predict.

2 Management Information System

Management Information System (MIS) is one of the final steps in an analysis as it is the tool that supports the decision makers of company or a project [3]. The main purpose of a system as this is to use the outcome of an analysis and provide insights based on the pre-process, uncertainty, filtering, storing, and most importantly modeling

that was done on the data [1]. Thanks to those information analysts can provide to the decision makers the data they need to make any decision they need to make [4].

In Fig.1. a MIS menu system is presented, which is divided in three parts. The first part is where the user can insert new data, display it so he can gain insight and lastly start the prep-processing. After that the user can start taking the necessary actions needed to start build a model. Finally, on the last part he can execute the model built and can print the results.

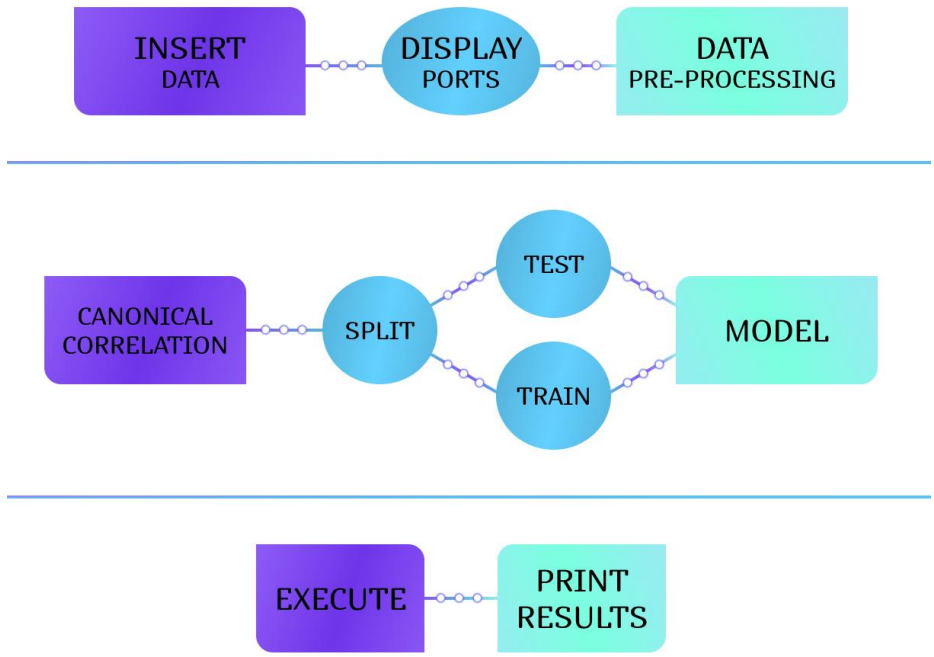

Fig. 1. MIS Menu Design

A brief description of each of the buttons in the menu of MIS.

- *Display Reports.* This step displays the data, the variables, the missing values, and many more useful information to gain insight on the data.
- *Data Preprocessing.* This step allows the owner to start the pre-process, meaning removing missing values, dealing with uncertainty, renaming variables, or even deleting variables because they are not needed.

- *Split for Re-evaluation.* This is the point in the system where the owner can create or upgrade an already existing model. Given that an improvement can be made, anew model created by splitting the data set and training a new model with better accuracy.
- *Execute.* This step is the point where most of the code is going to be executed.
- *Print Results.* As the final step, the owner can print the result of his analysis, meaning he can any prediction that the model made or even useful statistics revolving around the data, this step is the main output of the whole code and give us most of our results.

3 Data Analysis

3.1 Data Presentation

This study will focus on a data set containing statistics of dataset of basketball players through Seasons 1999-2020 and from 49 Leagues [5]. The data set is a csv file, and it is consisted of 53949 row and 34 variables as shown in **Fig. 2.** along with a data set sample. For this study only 19 variables are going to be used. [Appendix A]

Total rows: 53949 Total columns: 34

	League	Season	Stage	Player	Team
1	NBA	1999 - 2000	Regular_Season	Shaquille O'Neal	LAL
2	NBA	1999 - 2000	Regular_Season	Vince Carter	TOR
3	NBA	1999 - 2000	Regular_Season	Karl Malone	UTA
4	NBA	1999 - 2000	Regular_Season	Allen Iverson	PHI
5	NBA	1999 - 2000	Regular_Season	Gary Payton	SEA
6	NBA	1999 - 2000	Regular_Season	Jerry Stackhouse	DET
7	NBA	1999 - 2000	Regular_Season	Grant Hill	DET
8	NBA	1999 - 2000	Regular_Season	Kevin Garnett	MIN
9	NBA	1999 - 2000	Regular_Season	Michael Finley	DAL
10	NBA	1999 - 2000	Regular_Season	Chris Webber	SAC

Fig. 2. Sample of Original Data set

Description of the Variables.

- **League** is the name of the League each observation/player belongs to.
- **Season** is the year in which the statistics of each player were documented.
- **Stage** describes the type of stage in which the statistics of each player were documented.
- **Player** is the full name of each basketball player.
- **Team** is the name of the team each basketball player belongs to.

- **GPP** is the number of games each player had played professionally.
- **MIN** is the number of minutes each player had played.
- **FGM** is the number of field goals each player made.
- **TPM** is the number of three pointers each player made.
- **TPA** is the number of three pointes each player attempted.
- **FTM** is the number of free throws each player made.
- **FTA** is the number of free throws each player attempted.
- **TOV** is the number of turn overs each player made.
- **PF** is the number of personal fouls each player made.
- **ORB** is the number of offensive rebounds each player made.
- **DRB** is the number of defensive rebounds each player made.
- **REB** is the total number of rebounds per player.
- **AST** is the number of assists per player.
- **STL** is the number of steals per player.
- **BLK** is the number of blocks per player.
- **PTS** is the number of point each player scored.
- **birth_year** is the year each player was born.
- **birth_month** is the month is player was born.
- **height** is the height of each player in inches.
- **height_cm** is the height of each player in centimeters.
- **weight** is the number of pounds each player weighs.
- **weight_kg** is the number of kilos each player weighs.
- **nationality** is the nation of which each player belongs.
- **high_school** is the high school of which each player graduated from.
- **draft_round** is the number of draft rounds.
- **draft_pick** is the number of draft picks.
- **draft_team** is the name of the draft team.

3.2 Data Import

To start with we created the first dataset from the .csv file by using the PROC IMPORT statement along with a FILE statement which identifies an external file to read, the path of the file is needed [6]. In **Fig. 3**. we can see how the code looks.

The DELIMITER option is used to specify the delimiter between values. In this case we use "**;**" as a separator. The OUT option identifies the output SAS data set with either a one or two-level SAS name (library and member name). If the specified SAS data set does not exist, the IMPORT procedure creates it. By default, the PROC IMPORT procedure uses either the USER library (if assigned) or the WORK library (if USER is not assigned) which is what happened in our case. Furthermore, "DBS" specifies the type of the data that we want to import. For this study our original data set was a ".csv" file and we let SAS know it with this option. Finally, the REPLACE option overwrites an existing SAS data set.

```
/*CSV LOAD  */
proc import file='/home/u59859401/sasuser.v94/Task2/players_stats_by_season_full_details.csv'
    out=task2
    dbms=csv
    replace;
    delimiter=";";
run;
/*CSV LOAD  */
```

Fig. 3. Proc Import

3.3 Data Pre-Processing

In the next step as we can see in **Fig. 4,** we start the preprocessing step by "dropping" from the data set whatever variable we think is not going to be useful in our study and by renaming some of those that we are going to use for better handling of the data set.

```
/* Data Pre-Processing */
data task2keep;
    set task2;
    drop draft_team nationality high_school birth_year birth_month birth_date
    League Stage Player Team height season draft_round draft_pick weight;
    rename height_cm=HEI weight_kg=WEI draft_round=DRR draft_pick=DRP;
run;
/* Data Pre-Processing */
```

Fig. 4. Removing Variables

Now we can inspect our data set for all the variable name, length, type, format and informat using the "proc contents" command.

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Informat
14	AST	Num	8	BEST12.	BEST32.
16	BLK	Num	8	BEST12.	BEST32.
12	DRB	Num	8	BEST12.	BEST32.
4	FGA	Num	8	BEST12.	BEST32.
3	FGM	Num	8	BEST12.	BEST32.
8	FTA	Num	8	BEST12.	BEST32.
7	FTM	Num	8	BEST12.	BEST32.
1	GPP	Num	8	BEST12.	BEST32.
18	HEI	Num	8	BEST12.	BEST32.
2	MIN	Num	8	BEST12.	BEST32.
11	ORB	Num	8	BEST12.	BEST32.
10	PF	Num	8	BEST12.	BEST32.
17	PTS	Num	8	BEST12.	BEST32.
13	REB	Num	8	BEST12.	BEST32.
15	STL	Num	8	BEST12.	BEST32.
9	TOV	Num	8	BEST12.	BEST32.
6	TPA	Num	8	BEST12.	BEST32.
5	TPM	Num	8	BEST12.	BEST32.
19	WEI	Num	8	BEST12.	BEST32.

Fig. 5 Inspection of Data Set

Last step of pre-processing our data by searching if there are any missing values, after that we try to delete them. By using the ‘’PROC MEANS” command alongside the option “N NMISS” we receive the output seen in **Fig. 6**

Variable	N	N Miss
GPP	53949	0
MIN	53949	0
FGM	53949	0
FGA	53949	0
TPM	53949	0
TPA	53949	0
FTM	53949	0
FTA	53949	0
TOV	53949	0
PF	53949	0
ORB	53949	0
DRB	53949	0
REB	53949	0
AST	53949	0
STL	53949	0
BLK	53949	0
PTS	53949	0
HEI	53875	74
WEI	49385	4564

Fig. 6 Missing Values

Now we know how many observations each variable has and more importantly how many missing values per observation each variable has. We observe that our data

set has 74 missing values in the “HEI” variable and 4564 in the “WEI” variable.

Lastly, as shown in **Fig.7**, with the help of the ARRAY statement along with the TO-DO loop, we deleted the missing values on every column of the dataset.

```
/*DELETING OBSERVATIONS WITH MISSING VALUES*/
data task2keepclean;
    set task2keep;
    array var _numeric_;
    do over var;
        if missing(var) then delete;
    end;
run;
/*DELETING OBSERVATIONS WITH MISSING VALUES*/
```

Fig. 7 Deleting Missing Values

By checking again with “Proc Means” we can see that the data set has 49385 observations left

3.4 Correlation

As mentioned before the goal of this study is to make a model that predicts the variable “GPP” meaning, the number of games each player played. In order to do that in the most efficient way, we first check the correlation between the variable “GPP” and the rest variables of the data set using the “PROC CANCORR” procedure. This procedure is a way of inferring information from cross-covariance matrices and the syntax of it is as seen in **Fig. 8**

```
/*CANNONICAL CORRELATION */
proc cancorr data=task2keepclean all;
var GPP;
with MIN FGM FGA TPM TPA FTM FTA TOV PF ORB DRB REB AST STL BLK PTS HEI WEI;
RUN;
/*CANNONICAL CORRELATION */
```

Fig. 8 Proc Cancorr

As an output from this procedure, we gain a great number of tables containing greatly important statistical information, but for the purpose of this study we are concerned about the canonical correlation of “GPP” with the rest of the variables of the data set as seen in **Fig. 9**

Correlations Between the WITH Variables and the Canonical Variables of the VAR Variables	
	V1
MIN	0.9050
FGM	0.7714
FGA	0.7864
TPM	0.5007
TPA	0.5229
FTM	0.6390
FTA	0.6516
TOV	0.7399
PF	0.8872
ORB	0.5999
DRB	0.7448
REB	0.7266
AST	0.6102
STL	0.7101
BLK	0.5127
PTS	0.7646
HEI	0.0423
WEI	0.0700

Fig. 9 Canonical Correlation

From the table above we can observe that the variables “HEI” and “WEI” which represent height and weight respectively have and extremely low correlation with the variable “GPP”. Any correlation lower below 0.3 tells us that the variable should not be included in the model.

3.5 Splitting the Data Set and Modeling

Before we start the building of the model, we split the dataset “task2keepclean2” into a training dataset which we named it “train” and a testing dataset named “test”. To do that we used the “SURVEYSELECT” procedure with a sampling rate of 70% meaning that 70% of the base dataset is imported in the training dataset and the rest, which will be used for testing the model, is imported in the testing one [7]. With the METHOD option, we specified that the sampling method is the simple random sampling (SRS).

```
/*SPLITING*/
proc surveyselect data=task2keepclean2 rate=0.7
out= select outall
method=SRS;
run;
data train test;
    set select;
    if selected=1 then output train;
    else output test;
run;
/*SPLITING*/
```

Fig. 10 Splitting

As shown in the next figure **Fig. 11** we used “PROC GLMSELECT” on the “train” dataset” which performs an effect selection in the framework of general linear models. With this we can produce parameter estimates which help us build a better model. To add to, with the SELECTION option we specify the selection technique which is BACKWARDS. This selection technique begins with a model that contains all variables under consideration, then starts removing the least significant variables one after the other until no variables are left in the model. We also include the “details =steps” so we can see what the order of each variable removed was. After that, to build our model, we use “PROC REG”. We specify the PLOTS option in a way that requests to produce all the plots available even though we have almost 50000 observations. In the next statement MODEL, we specify the dependent variable GP and the regressor variables which are all estimated by the “PROC GLMSELECT”. The significance level entry of the variables (SLE) and the significance level stay (SLS) is set to zero. Lastly, we output the results in a new dataset named “Reg_stats0001” adding the predicted and residual columns “p_” and “r_” respectively.

```
/*LINEAR REGRESSION MODEL */
ods noproctitle;
ods graphics;
proc glmselect data=train outdesign(addinputvars)=Work.reg_design
        plots=(criterionpanel);
    model GPP=MIN FGM FGA TPM TPA FTM FTA TOV PF ORB DRB REB AST STL BLK PTS /
        showpvalues selection=backward

   (slstay=0.05 select=sl stop=sl) details=steps;
run;

proc reg data=Work.reg_design alpha=0.05 plots(only)=all PLOTS(MAXPOINTS=50000);
    ods select DiagnosticsPanel ResidualPlot RStudentByPredicted DFFITSPlot
        DFBETASPanel ObservedByPredicted;
    model GPP=&_GLSMOD /;
    output out=work.Reg_stats0001 p=p_ r=r_;
    run;
quit;
/*LINEAR REGRESSION MODEL */
```

Fig. 11 Linear Regression

In the next figure we can see all the estimates of the parameters that were created from "PROC GLMSELECT".

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	4.547137	0.067193	67.67	<.0001
MIN	1	0.030003	0.000283	106.06	<.0001
FGM	1	-0.008375	0.002778	-3.02	0.0026
FGA	1	-0.006673	0.001446	-4.62	<.0001
TPM	1	-0.057291	0.007148	-8.01	<.0001
TPA	1	0.024244	0.002979	8.14	<.0001
FTA	1	-0.024919	0.001036	-24.06	<.0001
TOV	1	-0.042960	0.002603	-16.51	<.0001
PF	1	0.159153	0.001751	90.88	<.0001
ORB	1	0.014044	0.002187	6.42	<.0001
DRB	1	-0.014562	0.001056	-13.78	<.0001
AST	1	0.004627	0.000964	4.80	<.0001
STL	1	-0.047834	0.002887	-16.57	<.0001

Fig. 12 Parameter Estimates

"PROC REG" selected our model in three steps, we can see in **Fig. 12** that which variables met the $p < 0.1$ level of significance thus except "FGA", which did not pass the entry check into the model. We can conclude that almost all the independent variables are significant to the GP.

Analysis of Variance					
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	12	10211085	850924	21437.0	<.0001
Error	34557	1371710	39.69414		
Corrected Total	34569	11582795			

Root MSE	6.30033
Dependent Mean	30.67718
R-Square	0.8816
Adj R-Sq	0.8815
AIC	161844
AICC	161844
SBC	127382

Fig. 13 Analysis of Variance

Judging by the values of Sum of Squares, Root MSE Adj-Sq we can see that the model might be a good fit for the dataset as the RSS measures the level of variance in the error term and it is not high enough to disqualify our model. Next, RMSE is a metric showing how much concentrated our data is around the best fit line. Furthermore, Adj-R-Squared is the adjusted percentage of the independent variable variation that is explained by the linear model, a value of 0.8815 gives us the idea that this model can be pretty accurate. Even though most of the statistic measures indicate that we may have built a good model there seems be a linear relationship in the residual plot, see **Fig.14**, which in turn makes suspect that linear regression might not be the best method to use to predict the variable "GPP". More information was gained during the testing phase.

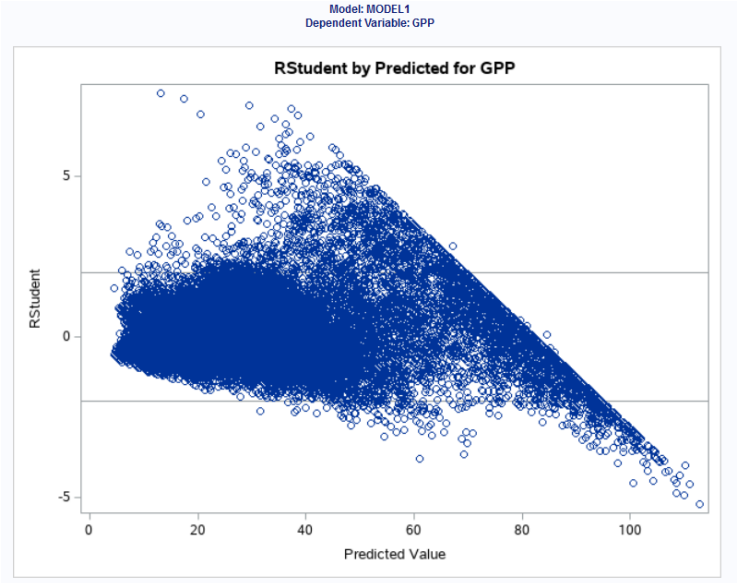

Fig. 14 Predicted for GPP

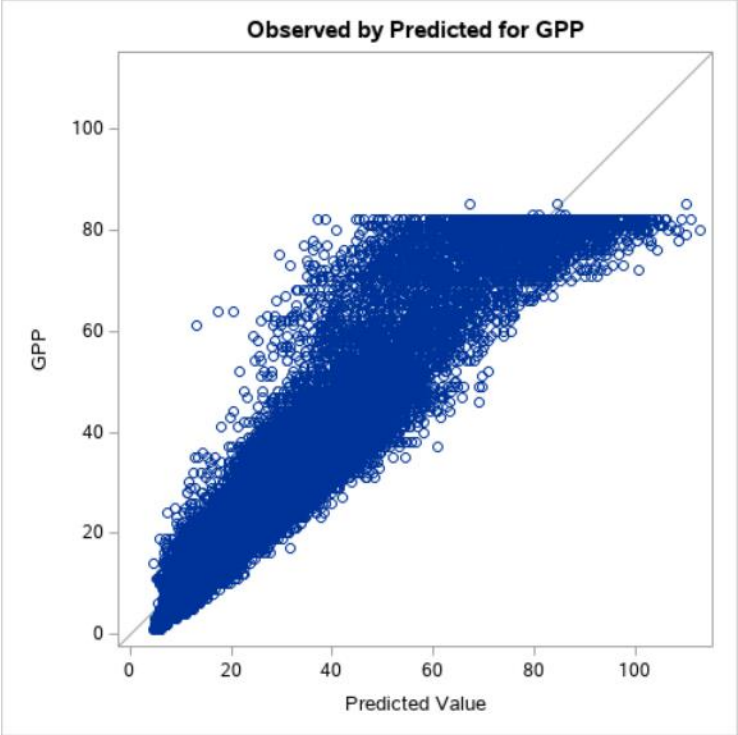

Fig. 15 Observed for GPP

4 Efficient Programming

An efficient code has the least run time while using the least resources. Minimizing the use of those resources, we can greatly improve the performance thus, programming efficiency is achieved. CPU Time, Data Storage, Input/Output (I/O) and Memory Usage are some of the resources of a system [1].

Programming languages are very different in nature from one another. Not just in terms of using a syntax or specific way of coding, but also where power consumption is concerned. Some techniques in SAS can approach this ideal scenario, either by using specific statements an option or even modifying existing ones which come be proven to be a great challenge to an analyst.

In this study, in order to improve our code and make it more efficient we use the SASFILE statement which loads our data set into a library.

```
NOTE: There were 53949 observations read from the data set WORK.TASK2KEEP.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           2:15.95
      user cpu time       2:15.82
      system cpu time     0.13 seconds
      memory              2584.96k
      OS Memory           32688.00k
      Timestamp           26/04/2022 04:39:01 π.μ.
      Step Count                        154  Switch Count  2
      Page Faults                       0
      Page Reclaims                     252
      Page Swaps                        0
      Voluntary Context Switches        15
      Involuntary Context Switches      172
      Block Input Operations            0
      Block Output Operations           60816
```

Fig. 16 LOG 1

```
/*EFFICIENCY */
libname task2 '/home/u59859401/sasuser.v94/Task2';
sasfile task2keep load;
proc print data=task2keep;
run;
sasfile task2keep close;
/*EFFICIENCY */
```

Fig. 17 Efficient Programming

```
NOTE: There were 53949 observations read from the data set WORK.TASK2KEEP.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           2:14.78
      user cpu time       2:14.66
      system cpu time     0.13 seconds
      memory              986.37k
      OS Memory           39728.00k
      Timestamp           26/04/2022 05:02:14 π.μ.
      Step Count                        190  Switch Count  2
      Page Faults                       0
      Page Reclaims                     26
      Page Swaps                        0
      Voluntary Context Switches        15
      Involuntary Context Switches      163
      Block Input Operations            0
      Block Output Operations           60816
```

Fig. 18 LOG2

As shown above, when we write the code in a regular way, its produced log (see **Fig16.**) the metric “real time cpu” gives us a metric of 2:15.95, while the improved

version of the code and its respective log (see Fig.17.) we can see that the same metric has become2:14.78. In the improved version, we used the SASFILE statement to load the “task2keep” data set, in other words we transferred the data to memory, and we read that data from that location. As we can, the real time and user CPU time were both decreased 1.17 minutes and the memory greatly decreased from 2584.96k to 986.37k. By using the improved version of the code, we greatly reduced our system resources, for example memory, and also minimized the cpu real time metric for that part of the code.

5 UML Diagram – Code Flow Chart

In this section, a flow chart of the structure of the code is presented.

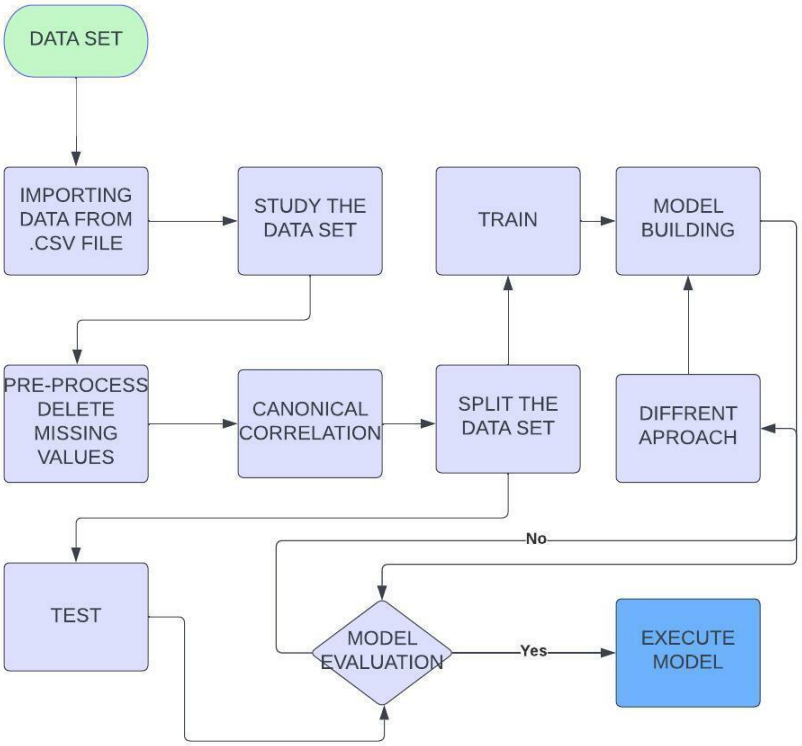

Fig. 19 Code Flow Chart

6 Conclusion

To sum up, we tried creating a MIS menu system which could correspond to the code we wrote. The goal of this was not only to give a more visual que to code but also to provide the user with the necessary interface so he can perform the analysis he wants. Next, we tried studying the correlation between variables so we can select the optimal regressors for out model. Our study revolved around the variable "GPP" which tells us how many games each player played, we tried to predict that value. Although we managed to create a relatively good model, the way the residuals were formed made us question our original approach. Further experimentation on the data set is need in order to find the optimal method for predicting the previously mentioned variable.

A second version of a part of the code was produced and we managed to achieve a better time of execution while saving a lot of memory. This was done with SASFILE statement which turned out to be a very important technique that can minimize the waste of the resources of a system. Last but not least, all the steps of the coding process were displayed in a flow diagram.

References

1. X. Wang and Y. He, "Learning from Uncertainty for Big Data," *Ieee Syst. Man Cybern. Mag.*, no. August, 2016.
2. J. Bendler, S. Wagner, T. Brandt, and D. Neumann, "Taming uncertainty in big data: Evidence from social media in urban areas," *Bus. Inf. Syst. Eng.*, vol. 6, no. 5, pp. 279–288, 2014
3. O'Brien JA, & Marakas, G.: Management Information Systems. 7th edn. Boston: Irwin McGraw-Hill, 1999.
4. J. Bendler, S. Wagner, T. Brandt, and D. Neumann, "Taming uncertainty in big data: Evidence from social media in urban areas," *Bus. Inf. Syst. Eng.*, vol. 6, no. 5, pp. 279–288, 2014.
5. https://www.kaggle.com/datasets/jacobbaruch/basketball-players-stats-per-season-49-leagues
6. SAS Institute Inc.: SAS/ETS® 9.1 User's Guide. Cary, NC: SAS Institute Inc. 2004.
7. Williams, G.: Descriptive and predictive analytics. In: Data Mining with Rattle and R, pp. 171-177, Springer, New York, NY 2011.

Appendix A Data Set

Total rows: 53949 Total columns: 34 Rows 1-100

	League	Season	Stage	Player	Team	GPP
1	NBA	1999 - 2000	Regular_Season	Shaquille O'Neal	LAL	79
2	NBA	1999 - 2000	Regular_Season	Vince Carter	TOR	82
3	NBA	1999 - 2000	Regular_Season	Karl Malone	UTA	82
4	NBA	1999 - 2000	Regular_Season	Allen Iverson	PHI	70
5	NBA	1999 - 2000	Regular_Season	Gary Payton	SEA	82
6	NBA	1999 - 2000	Regular_Season	Jerry Stackhouse	DET	82
7	NBA	1999 - 2000	Regular_Season	Grant Hill	DET	74
8	NBA	1999 - 2000	Regular_Season	Kevin Garnett	MIN	81
9	NBA	1999 - 2000	Regular_Season	Michael Finley	DAL	82
10	NBA	1999 - 2000	Regular_Season	Chris Webber	SAC	75
11	NBA	1999 - 2000	Regular_Season	Ray Allen	MIL	82
12	NBA	1999 - 2000	Regular_Season	Alonzo Mourning	MIA	79
13	NBA	1999 - 2000	Regular_Season	Tim Duncan	SAS	74
14	NBA	1999 - 2000	Regular_Season	Glenn Robinson	MIL	81
15	NBA	1999 - 2000	Regular_Season	Antoine Walker	BOS	82
16	NBA	1999 - 2000	Regular_Season	Shareef Abdur-Rahim	VAN	82
17	NBA	1999 - 2000	Regular_Season	Stephon Marbury	NJN	74
18	NBA	1999 - 2000	Regular_Season	Elton Brand	CHI	81
19	NBA	1999 - 2000	Regular_Season	Allan Houston	NYK	82
20	NBA	1999 - 2000	Regular_Season	Antonio McDyess	DEN	81

Appendix B SAS Code

```
/*CSV LOAD  */

proc import file='/home/u59859401/sasuser.v94/Task2/players_stats_by_season_full_details.csv'
    out=task2
    dbms=csv
    replace;
    delimiter=";";
run;
/*CSV LOAD  */
/* Data Pre-Processing */

data task2keep;
    set task2;
    drop draft_team nationality high_school birth_year birth_month
    birth_date League Stage Player Team height season draft_round draft_pick weight;
    rename height_cm=HEI weight_kg=WEI draft_round=DRR draft_pick=DRP;
run;
proc print data=task2keep;
run;
/* Data Pre-Processing */
 /*EFFICIENCY */
 libname task2 '/home/u59859401/sasuser.v94/Task2';
 sasfile task2keep load;
 proc print data=task2keep;
 run;
 sasfile task2keep close;
 /*EFFICIENCY */
 /*FINDING HOW MANY MISSING VALUES THERE ARE*/
 proc means data=task2keep N NMISS ;
 VAR GPP MIN FGM FGA TPM TPA FTM FTA TOV PF ORB DRB REB AST STL BLK PTS HEI WEI;
 RUN;
 /*FINDING HOW MANY MISSING VALUES THERE ARE*/
 /*DELETING OBSERVATIONS WITH MISSING VLAUES*/
 Data task2keepclean;
     set task2keep;
     array var _numeric_;
     do over var;
         if missing(var) then delete;
     end;
 run;
 /*DELETING OBSERVATIONS WITH MISSING VLAUES*/
```

```
/* CHECKING FOR ANY MISSING VALUES AGAIN */
proc means data=task2keepclean N NMISS ;
VAR GPP MIN FGM FGA TPM TPA FTM FTA TOV PF ORB DRB REB AST STL BLK PTS HEI WEI;
RUN;
/* CHECKING FOR ANY MISSING VALUES AGAIN */
/*CANNONICAL CORRELATION */
proc cancorr data=task2keepclean all;
var GPP;
with MIN FGM FGA TPM TPA FTM FTA TOV PF ORB DRB REB AST STL BLK PTS HEI WEI;
RUN;
/*CANNONICAL CORRELATION */
/* DROP HEI WEI */
Data task2keepclean2;
    set task2keepclean;
    drop WEI HEI;
run;
/* DROP HEI WEI */
proc surveyselect data=task2keepclean2 rate=0.7
out= select outall
method=SRS;
run;
data train test;
    set select;
    if selected=1 then output train;
    else output test;
run;
/*SPLITING*/

/*LINEAR REGRESSION MODEL */
ods noproctitle;
ods graphics;
proc glmselect data=train outdesign(addinputvars)=Work.reg_design
        plots=(criterionpanel);
    model GPP=MIN FGM FGA TPM TPA FTM FTA TOV PF ORB DRB REB AST STL BLK PTS /
        showpvalues selection=backward

   (slstay=0.05 select=sl stop=sl) details=steps;
run;

proc reg data=Work.reg_design alpha=0.05 plots(only)=all PLOTS(MAXPOINTS=50000);
    ods select DiagnosticsPanel ResidualPlot RStudentByPredicted DFFITSPlot
        DFBETASPanel ObservedByPredicted;
    model GPP=&_GLSMOD /;
    output out=work.Reg_stats0001 p=p_ r=r_;
    run;
quit;
/*LINEAR REGRESSION MODEL */
```