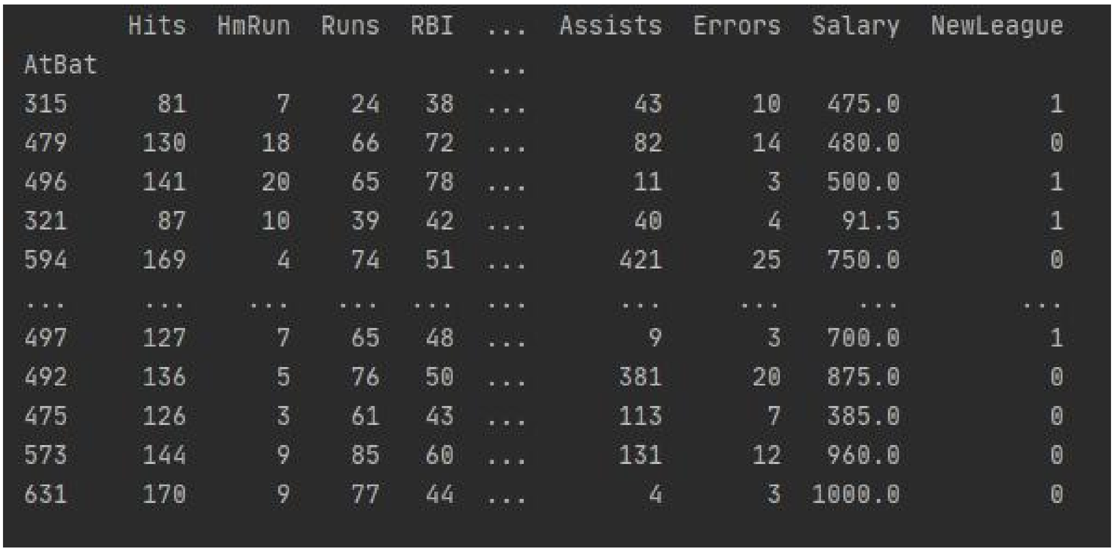
1. Is this a good method? Do you expect to obtain the true prediction error? Explain your answer.

No, I don’t think it’s a good method, because the first two steps in the cross-validation procedure question gave us were wrong. Firstly, finding predictors based on correlation with response variable, which means calculating correlation based on all sample(including training dataset and test dataset). Secondly, using the correlated predictor to build model, which might lead to some overlapping data between the training set and the test set when building model. Therefore, these predictors have already seen the left out samples, because model has already gotten some information about test data.

Thus, if we want to expect to obtain the true prediction error, the correct cross-validation procedure is:
Firstly, randomly dividing all sample into to K cross-validation folds.
Secondly, finding a subset of good predictors for each fold k = 1,2,...,K.
Thirdly, using subset of predictors of each fold to build a multivariate classifier.
Finally, using the classifier to predict the class labels for the samples in fold k.

2.
(a) (5%) Load the data-set and replace all categorical values with numbers. (You can use the LabelEncoder object in Python).



```
       Hits  HmRun  Runs  RBI  ...  Assists  Errors  Salary  NewLeague
AtBat                          ...
315      81      7    24   38  ...       43      10   475.0          1
479     130     18    66   72  ...       82      14   480.0          0
496     141     20    65   78  ...       11       3   500.0          1
321      87     10    39   42  ...       40       4    91.5          1
594     169      4    74   51  ...      421      25   750.0          0
...     ...    ...   ...  ...  ...      ...     ...     ...        ...
497     127      7    65   48  ...        9       3   700.0          1
492     136      5    76   50  ...      381      20   875.0          0
475     126      3    61   43  ...      113       7   385.0          0
573     144      9    85   60  ...      131      12   960.0          0
631     170      9    77   44  ...        4       3  1000.0          0
```

(b) (5%) Fit linear regression and report 10-Fold Cross-Validation mean squared error.
By using mean_squared_error() function, the mean squared error of each fold are:
[111269.81, 36363.7232, 167330.3962, 90937.1291, 81190.1761, 62390.5722, 290085.8707, 140067.0647, 121352.4346, 103344.5557]

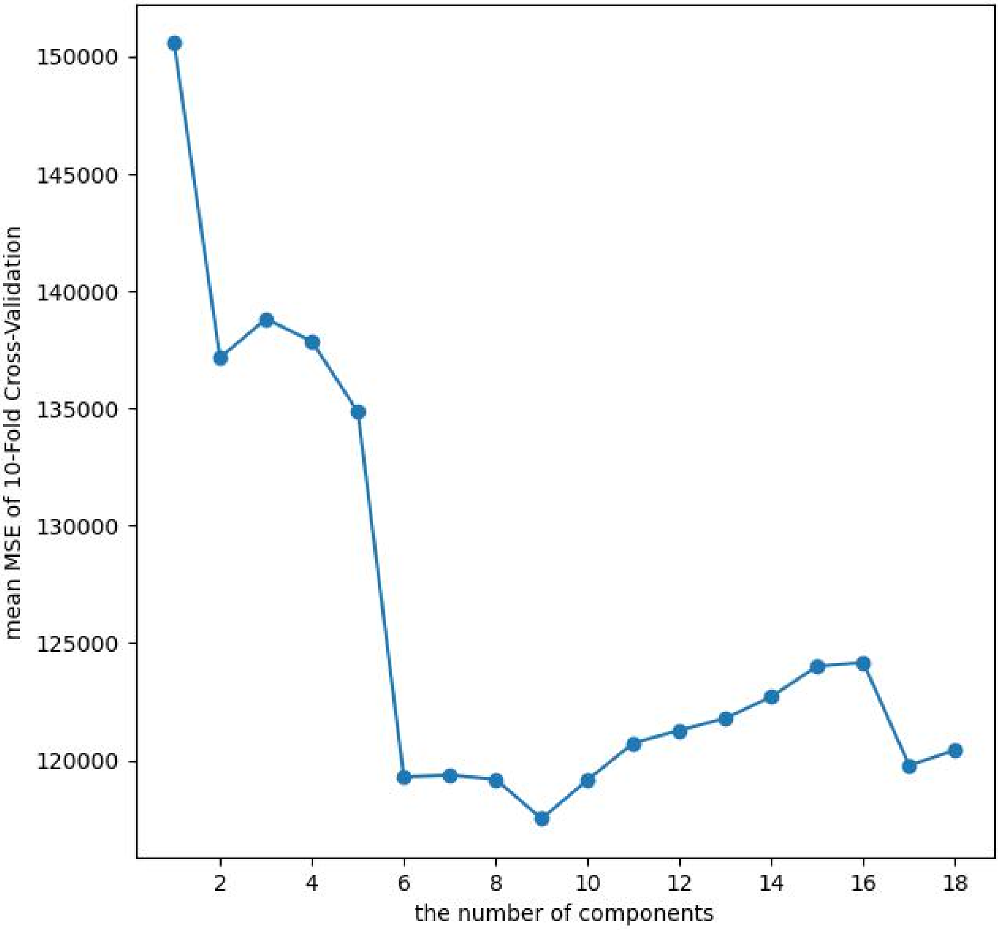
Hence, by using np.mean() function, 10-Fold Cross-Validation mean squared error is:

120433.17324999999

(c) (10%) Apply Principal Component Regression (PCR) with all possible number of principal components. Using the 10-Fold Cross-Validation, plot the mean squared error as a function of the number of components and determine the optimal number of components.

By applying PCR, the MSE of all possible number of principal components are: [150569.36106000002, 137160.59219000002, 138818.31405, 137859.83158, 134850.44520000002, 119293.94814000002, 119373.06041, 119191.51310000001, 117510.17512, 119163.00718999999, 120743.09962000002, 121283.62787000001, 121795.73466999999, 122715.94105, 124018.17003, 124174.73026000001, 119782.82264999999, 120433.17324999999]

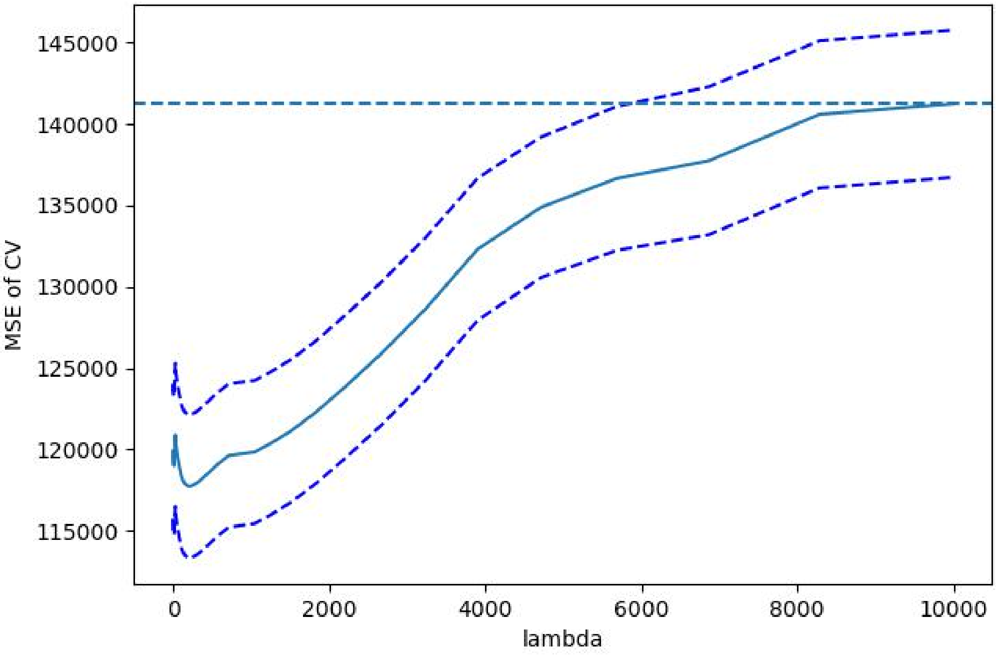
The plot of MSE of the number of components is:



From the plot of MSE of the number of components I listed above, we could find that the optimal number of components is 9, because the value of MSE of it is the lowest.

(d) (10%) Apply the Lasso method and plot the the 10-Fold Cross-Validation mean squared error as a function of λ. Determine the best λ and the corresponding mean squared error.
The best λ is 193.06977288832496 and and the corresponding mean squared error of it is 117751.00213726873.

I also plot the relationship between λ and their corresponding mean squared error:



3.Discuss the time complexity of your method in terms of n, e.g. is it O(n), O(ln(n)), etc. Give a short explanation (at most 2 sentences) for your answer.

Because the pmf formula is discrete, it uses inverse-transform method in discrete distribution to generate a random variable from the discrete pmf.

The time complexity of this method is O(1). That is because this method requires to convert pmf into commutative distribution function(CDF), and then we could get inverse function of the CDF. Because using x to generate X only needs one step, as computing the smallest integer that value is larger than x.

4. Show how to generate a random variable X ~ f(x).

Firstly, converting pdf into CDF:

$$\int_0^t 3(1-t)^2\,dt = 1-(1-t)^3$$

Secondly,computing inverse CDF:

(1) Original CDF: $y = 1-(1-x)^3$

(2) Inverse CDF: $x = 1-(1-y)^3$

Thirdly, generating random variables by Inverse CDF:

$$y = 1-\sqrt[3]{(x-1)}$$

5.

Firstly, generating 10000 random variable by X~f(x).
Then, computing average Y (expectation Y).
In Monte Carlo, we generally use an unbiased estimator of ℓ -- the sample mean:

$$\hat{\ell} = \frac{1}{N}\sum_{i=1}^{N} H(X_i),$$

where $X_1,\ldots,X_N \overset{iid}{\sim} f(x)$.

After that, computing 95% confidence interval.

By the central limit theorem, $\hat{\ell}$ has a N(ℓ, $\frac{S^2}{N}$) distribution.

Despite that the the σ^2 is generally unavailable, it can be estimated from Monte Carlo simulation via a calculation of a sample variance:

$$S^2 = \frac{1}{N\text{-}1}\sum_{i=1}^{N}(H(X_i)-\hat{\ell})^2 \quad (\text{N}\to\infty, S^2 \to \sigma^2)$$

Thus, an approximate 95% interval for ℓ is equal to:

$$(\hat{\ell} \pm z_{0.975}\frac{S}{N})$$

Consequently, the 95% confidence interval is (-189.118990646853, -189.654984240872).

6.

(a) Find the squared coefficient of variation CV^2 of Z.

Firstly, calculating Var(Z):
Generally, we couldn't know Var(Z) , so we could estimate it empirically using

$$Var(Z) \equiv S^2 = \frac{\ell_\gamma(1-\ell_\gamma)}{N}$$

Secondly, calculating E(Z):

Clearly, $\hat{\ell}_\gamma$ is an unbiased estimator:

$$E(Z) = E(\hat{\ell}) = E(\frac{1}{N}\sum_{i=1}^{N} Z_i) = \frac{1}{N}\sum_{i=1}^{N} EZ_i = \ell_\gamma$$

Because $CV^2 = Var(Z)/(E[Z])^2$, therefore, $CV^2 = \frac{\frac{\ell_\gamma(1-\ell_\gamma)}{N}}{\ell_\gamma^{\ 2}}$.

(b) Find the relative error of the estimator $\hat{\ell}_\gamma$ in terms of N and ℓ_γ.

If we want to find the relative error of the estimator $\hat{\ell}_\gamma$, we just need to calculate CV according to the result of question (a).

Therefore, the relative error of the estimator $\hat{\ell}_\gamma$ is:

$$CV = \sqrt{CV^2} = \frac{\sqrt{\frac{\ell_\gamma(1-\ell_\gamma)}{N}}}{\ell_\gamma}$$

(c) Prove that the CMC estimator is not logarithmically efficient.

From question (b), we could know that Z obeys Bernoulli Distribution. Therefore, according to Bernoulli's expectation formula, $E(Z^2) = \ell_\gamma$ and $(E[Z])^2 = \ell_\gamma^{\ 2}$.

Further, L'Hospital's Rule indicates that in this case, the limitation can be calculated by the differentiating both numerator and denominator.

Therefore, $\lim_{\gamma\to\infty}\frac{\ln E(Z^2)}{\ln(E[Z])^2} = \lim_{\gamma\to\infty}\frac{\ln \ell_\gamma}{\ln \ell_\gamma^{\ 2}} = \lim_{\gamma\to\infty}\frac{1}{2} = \frac{1}{2} \neq 1$. This means that the CMC estimator is not logarithmically efficient.