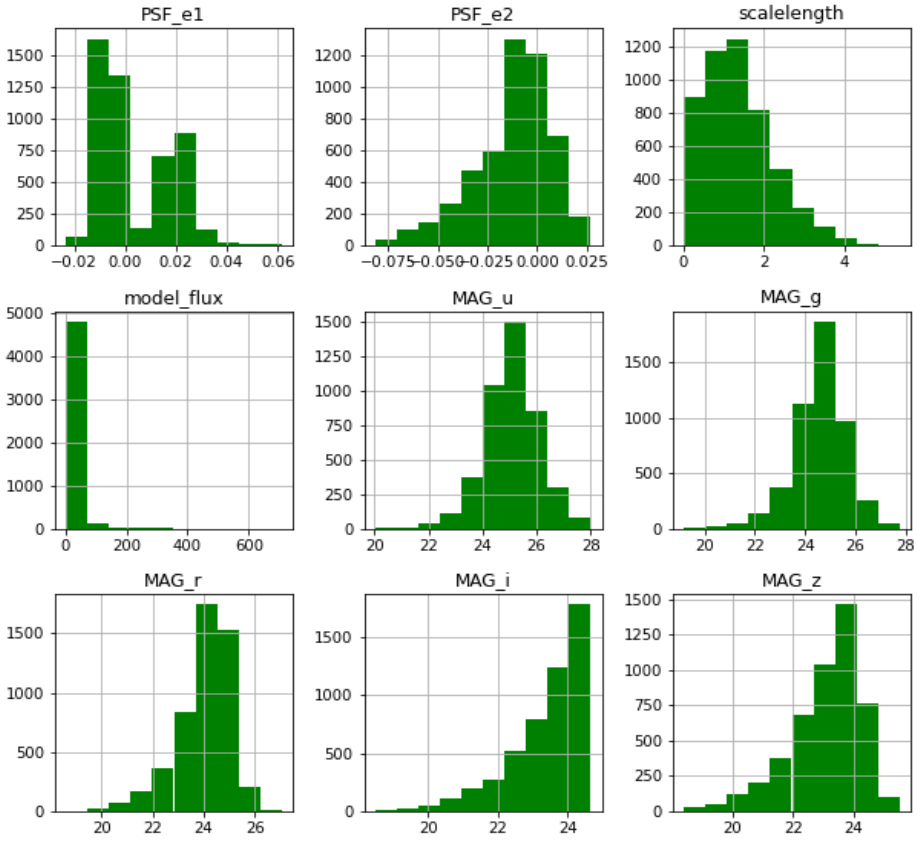
1. Your data set profile. Did you find any patterns in the missing data? Would you describe them as MCAR, MAR, or NMAR?

Answer:

- ✓ Data set profile:
  - Mean, Minimum, Median and Max Values

	mean	**min**	**median**	**max**
PSF_e1	0.003734	-0.0233	-0.0039	0.0617
PSF_e2	-0.01191	-0.0817	-0.009	0.0272
Scalelength	1.349806	0	1.24925	5.381
Model_flux	16.53538	1.6	6.6	708.4
MAG_u	25.09112	20.0114	25.1144	28.0086
MAG_g	24.5878	19.176	24.67415	27.7597
MAG_r	24.00826	18.6185	24.24045	27.0513
MAG_i	23.40669	18.4632	23.7177	24.6617
MAG_z	23.07616	18.3907	23.3323	25.5219

  - Histogram for the 9 features



- Number of missing values

Feature	Number of missing values
PSF_e1	0
PSF_e2	0
Scalelength	0
Model_flux	0
MAG_u	693
MAG_g	144
MAG_r	24
MAG_i	2
MAG_z	157

- ✓ When checking the missing data, I can see a correlation between MAG_g and MAG_U. The places where MAG_U is missed most of the corresponding MAG_g values are also missed. I think this can be considered as MAR (Missing At Random).
- ✓ Also, we can see that there are only 2 missing values in MAG_i, so I think this can be considered as MCAR (Missing Completely At Random). These missing values don't seem to have any correlation with other missing values. There is only one entry in the total dataset where all the 5 MAG features are missing. So, this can be a complete random miss (MCAR).

2. Description of the classifier and its parameter values.

Answer:

- ✓ I have used support vector machine(SVM) as my classifier which is based on libsvm.
- ✓ Parameters:
  - **C – value:** I have defaulted it to "1.0". This is the regularization parameter.
  - **Kernel:** By default, the kernel for the SVM will be "rbf" kernel. Based on the data provided in the assignment, I have marked the kernel as "linear" as it a linear classification problem.
  - **Random state:** To generate consistent results when the algorithm is running multiple times, we should not have any shuffling of the data. Hence, I have set the random state as "40" and used the same random state when running the experiments multiple times.

3. Description of the new method (E) that you developed/used.

Answer:

- ✓ The first method I tried which I reported in the train_eval.py and will be reporting in question 4 is "Iterative Imputer" method which happened to be my best method.
- ✓ In method (D), to use "average value" (mean) for the feature, I have used Simple Imputer with strategy as "mean". Any strategy of Simple Imputer typically fills the missing values by observing the other values in the same column. So, I want to try a

method which correlates with other features (columns) as well for filling the missing values. Hence, I selected the "Iterative Imputer" method.

- ✓ It works in a round-robin fashion by interacting with the other features. This approach allows for the use of previously imputed values as a component of a model to predict upcoming features because each feature is imputed sequentially, one after the other.
- ✓ I have set the random state as "40" (some random number) to get consistent results when running the algorithm multiple times.
- ✓ First, I have set only the random state and didn't make any changes to other parameters (default values). I achieved an accuracy of "78.10%" which is almost equal to the accuracy of method (C) but still a little lower value. Then I started tuning all the parameters.
- ✓ By default, the "imputation order" is "ascending" – the features will be imputed in the order of features with fewest missing values to most. I have modified this parameter to "descending" – the features will be imputed in the order of features with most missing values to fewest. I achieved a better accuracy of "79.05%" which is the best accuracy among all the methods.
- ✓ Also, I have tried other methods like KNN Imputer, Linear Interpolation with forward direction. I could achieve an accuracy ~ 78.2 but it is lower than method (C).

4. Test set classification accuracy using each of the missing value methods (A-E). Include two tables (full test set and those with missing values only).

Answer:

Method	Full test set accuracy	Missing test set accuracy
A (Abstention)	49.45%	0%
B (Predict Majority Class)	60.09%	40.77%
C (Omit Features)	78.8%	N/A
D (Impute with Avg Value)	70.75%	51.70%
E (Iterative Imputer)	79.05%	71.84%

- ✓ For missing test set, for method A, since the method is about not classifying items with missing values, in this case all items are considered as abstentions. Hence all the abstentions count as errors. The final accuracy will be 0.

5. Discussion/comparison of results. Which method do you recommend and why?

Answer:

- ✓ I observed that not classifying the items with missing values is definitely not a good solution as the most important information can be missed from those particular values.
- ✓ Method (B) also is same as above. Predicting the majority class may not be the correct solution, as I think there can be chances where the missing values can be from the class other than the one which has majority. Then our approach is not a valid one. This approach is kind of introducing bias to our algorithm.

- ✓ To some extent, I felt method (C) is good enough, but it has its own drawbacks. When we omit the features itself, there is a chance that we will lose valuable information. In this case, we can see that out of 9 features – since 5 of the features have at least one missing value, I have omitted all the 5 features (columns). Those 5 features happened to be magnitude which is definitely an important feature. So, although I achieved good accuracy, I still say that this is not an accurate way of prediction.
- ✓ Compared to methods A, B, C I feel definitely method (D) is a good one as we are not dropping any columns, not populating the missing labels with majority class (bias). Instead, we are trying to fix the missing value by some method. So, although the accuracy is not better than method (C), still I feel this is a good approach. The only concern with this approach is we are purely relying(mean) on the other values of the "same" column to populate the missing values but in reality, the missing values can have correlation between the other features as well.
- ✓ To my understanding, method (E) is the best method and I would recommend it. As I mentioned above with method (D), the drawback in method (D) can be fixed to some extent by following approaches like Iterative Imputer, KNN Imputer where the missing values are populated based on the correlation and understanding of other features as well. That is one of the reasons I feel that I am able to achieve the best accuracy among all the methods. With further fine tuning of parameters or by following similar approaches accuracy can be improved further.

6. Show your sky object's feature vector (plus label). What result did you get when classifying your own sky object? Was it classified correctly or not?

Answer:

- ✓ Featured vector of my own sky object:

Id	W2p1p2_173609
Pos	134.1612391999998 -1.9742479859999933
CLASS_STAR	0.0538558
PSF_e1	-0.0002
PSF_e2	-0.003
Scalelength	1.2711
Model_flux	6.9
MAG_u	24.8403
MAG_g	23.8874
MAG_r	23.6899
MAG_i	23.6647
MAG_z	23.2685

Label: Galaxy – Since the "CLASS_STAR" is not >= 0.5

- ✓ When I classified this data with my classifier, it was classified correctly. There are no missing values in the featured vector, so no additional method is applied. The result from my code of classifier is as below.

"The prediction of my own sky object is: [False] : Not a Star - it is a galaxy."

- ✓ Yes, my own sky object was classified correctly.

7. We already know the class of these sky objects. What is the utility of training a model to make predictions on this data?

   Answer:

   Although we knew the class of the sky objects, we still trained a model and built an algorithm for prediction. This is because the given dataset can be a small portion of a research study and with the field of astronomy there are higher chances to generate such data everyday in the future. So, I think training a model to make predictions on such data will help for future forecasting.