

**Introduction**

If one eats healthy foods, one will remain in good health and will not need to see the doctor often but if one doesn't do this, he will surely need to see a doctor much often. The average consultation fees of doctors are between Rs 300–700. This is after 10 to 12 years of training. When you compare the fees with what people in other professions charge, it is very reasonable as even an electrician or plumber in big cities charges 300 to 500 Rs as service charges.

But sometimes, there have been situations where we go to a doctor in an emergency and find that the consultation fees are too high. As data scientists, we all should do better. As we already have data that record important details about a doctor, so today we get to build a model to predict the doctor’s consulting fee.

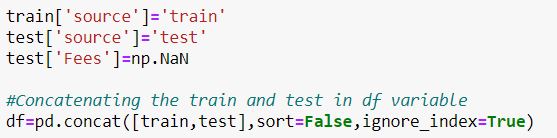
# Dataset Overview

Data consists of training and test sets.

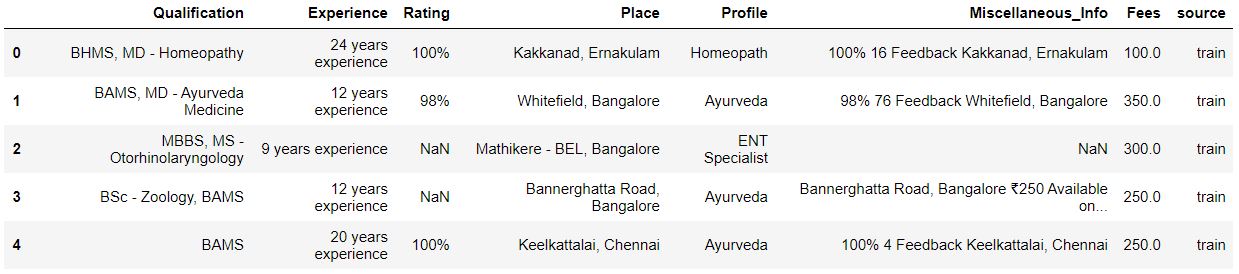
Size of training set: 5961 records

Size of test set: 1987 records

We will combine both these sets into one data frame for better data analysis and engineering. For combining train and test sets we create another column 'source', so that we can differentiate which rows belong to train and which to test and also add the 'Fees' column in the test dataset and initialize it with NaN values. The sole purpose of adding this column is to make the same number of columns in the train and test set so these two can be concatenated.



After combining, the new data set look like this.



**Data Description and Analysis**

Qualification: Qualification and degrees held by the doctor

Experience: Experience of the doctor in number of years

Rating: Rating given by patients

Profile: Type of the doctor

Miscellaneous Info: Extra information about the doctor

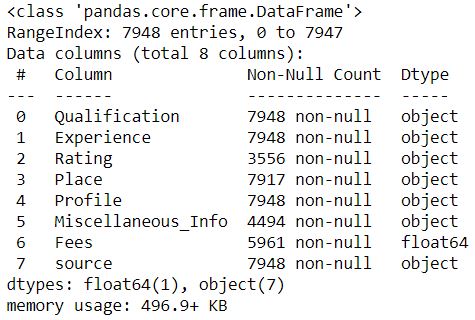
Fees: Fees charged by the doctor (Target Variable)

Place: Area and the city where the doctor is located.

Understanding the data is a must before its manipulation and analysis.

**Data structure**





1. Dataset has 7948 rows and 8 columns.
2. All the columns are of object datatype except Fees column that is of float type.
3. There are null values present in Rating, Place and Miscellaneous Info column.

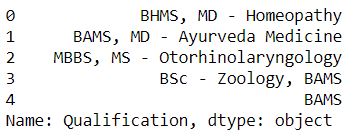
**Data Preparation**

As the data in the columns is in raw form. We will try to modify and clean it so that

all outdated or incorrect information is gone – leaving us with the highest quality information. Proper data preparation allows for efficient analysis - it can eliminate errors and inaccuracies that could have occurred during the data gathering process and can thus help in removing some bias resulting from poor data quality.

Qualification





This column contains all the qualifications of a doctor, separated by commas.

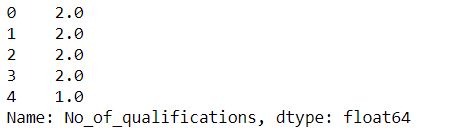




As there are so many unique values so it is difficult to find any pattern in qualification of a doctor with fees. So instead of qualifications, we dig out no. of qualifications each doctor has.

We count the number of qualifications for each doctor by separating the qualifications by a comma and then counting it and creating a new column "No\_of\_qualifications" from the qualifications column.

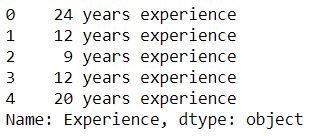




No of qualification column contains number of qualifications of the doctor in float type values.

Experience



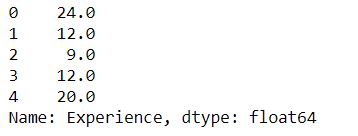


We only need the number of years of experience and not the words which follow. So we apply a lambda function to get only the digits from this column, convert them into float type

and store them back into the Experience column. We do this by extracting only the first two characters from experience.



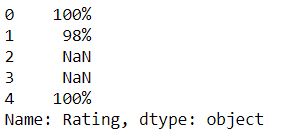




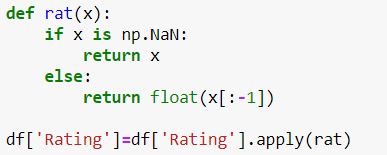
Now the experience column contains only the number of experiences in float values.

Rating

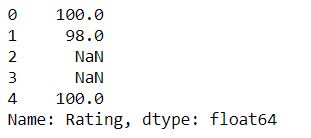




This column also has NaN values which we will take care of later. As for now, we need to remove the percent sign from ratings and convert it into numerical type data. We are going to do this by removing the last character from the rating and converting it to float datatype only if it is not a NaN value otherwise we keep the value as it is.

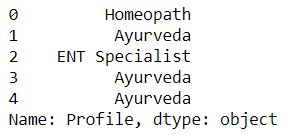






Profile

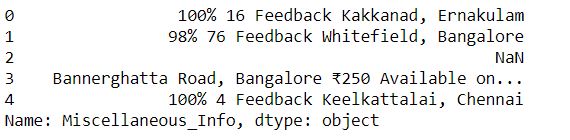




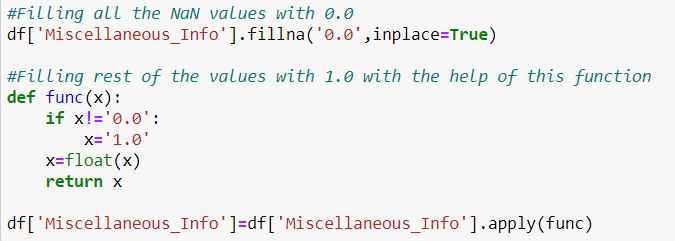
The profile column already seems to be clean, we just need to encode these Object type values into numeric type, which we will do it later in the pre-processing pipeline phase.

Miscellaneous\_Info

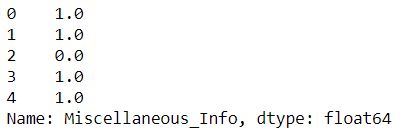




Since we cannot draw out any relevant information from this feature and as we have already seen above that there are a large number of NaN values present in this column. So we are going to replace all the NaN values with 0 and replace non-null values with 1 as Miscellaneous information provided is information in itself.

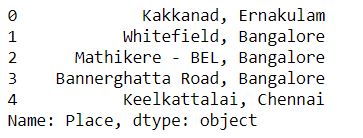






Place

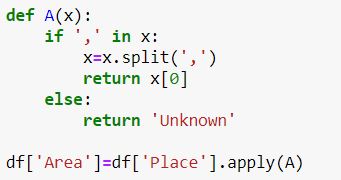




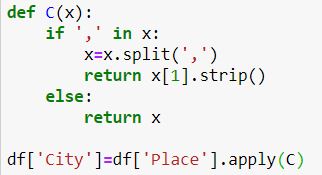
As we have seen earlier this column contains NaN values, so we will fill those NaN values with Unknown, Unknown place as this column contains the area and city to which the doctor belongs. We need to extract both this information as a doctor’s consultancy fees might also get affected by the area and city he/she practices.



Upon deep analysis, it is seen that in some places, the area of the place is missing and only the city is provided. We separate out the area and the city and store them in two different columns ‘Area’ and City. We do this by splitting the information by comma and putting the first word to Area column and second word to City column while, where do not find comma we put Unknown in the Area column and rest in the City column.

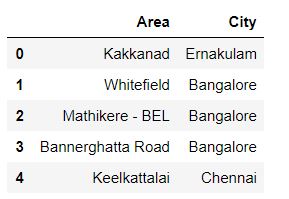


Above function helps in the extraction of area and appends Unknown where area is missing.



Above function helps in the extraction of city.

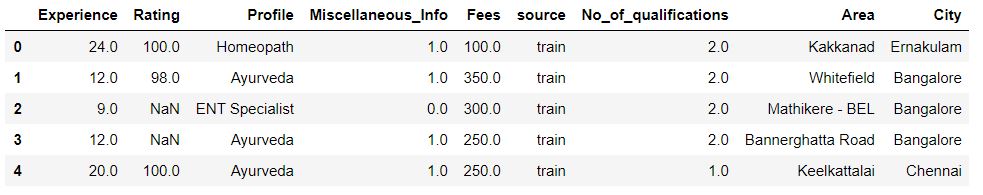




Now we have two columns, one which specifies the area and another the city, to which the doctor belongs.

We drop the unnecessary columns from which data has been already extracted.





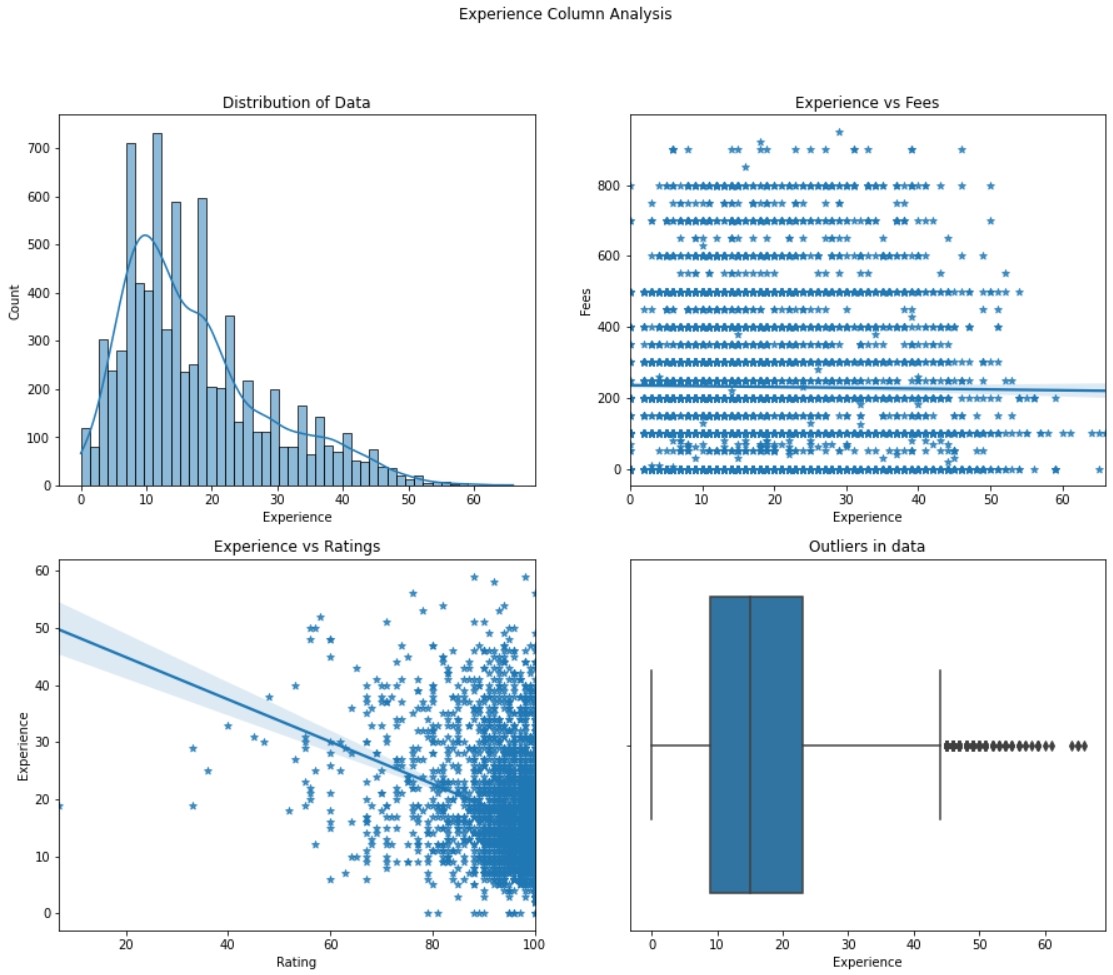
Data has been cleaned and gives much better information now.

**Exploratory Data Analysis**

We’ll go through each column iteratively and see which ones are useful for ML modeling later on. Some columns may need more pre-processing than others to get ready to use an algorithm.

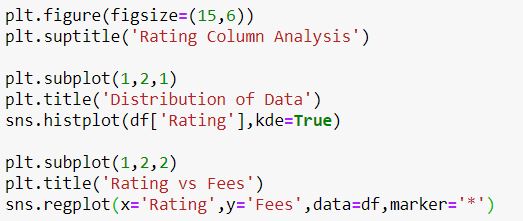
**Experience**

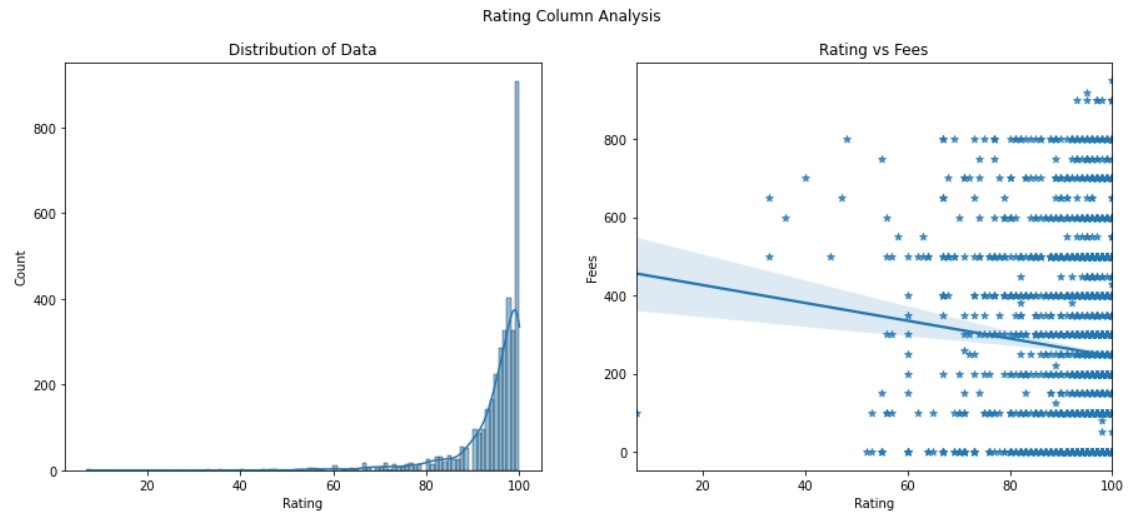




1. Experience of doctors starts from 0 years and goes beyond 60 years. The majority of the doctors have experience in the range of 5 years to 20 years. Experience data is Skewed towards the right.
2. Fees do not show much effect on Experience but there is a slight tilt in the intercept showing, fees increase with experience.
3. A high rating is shown in doctors with low experience. This reason could be because of the no. people who have given feedback. As rating also depends on the number of people who have given ratings. A doctor getting one 5 star rating will show higher ratings than a doctor who has got 4.5-star ratings from 100 people.
4. There are a large number of outliers present in this column.

**Rating**

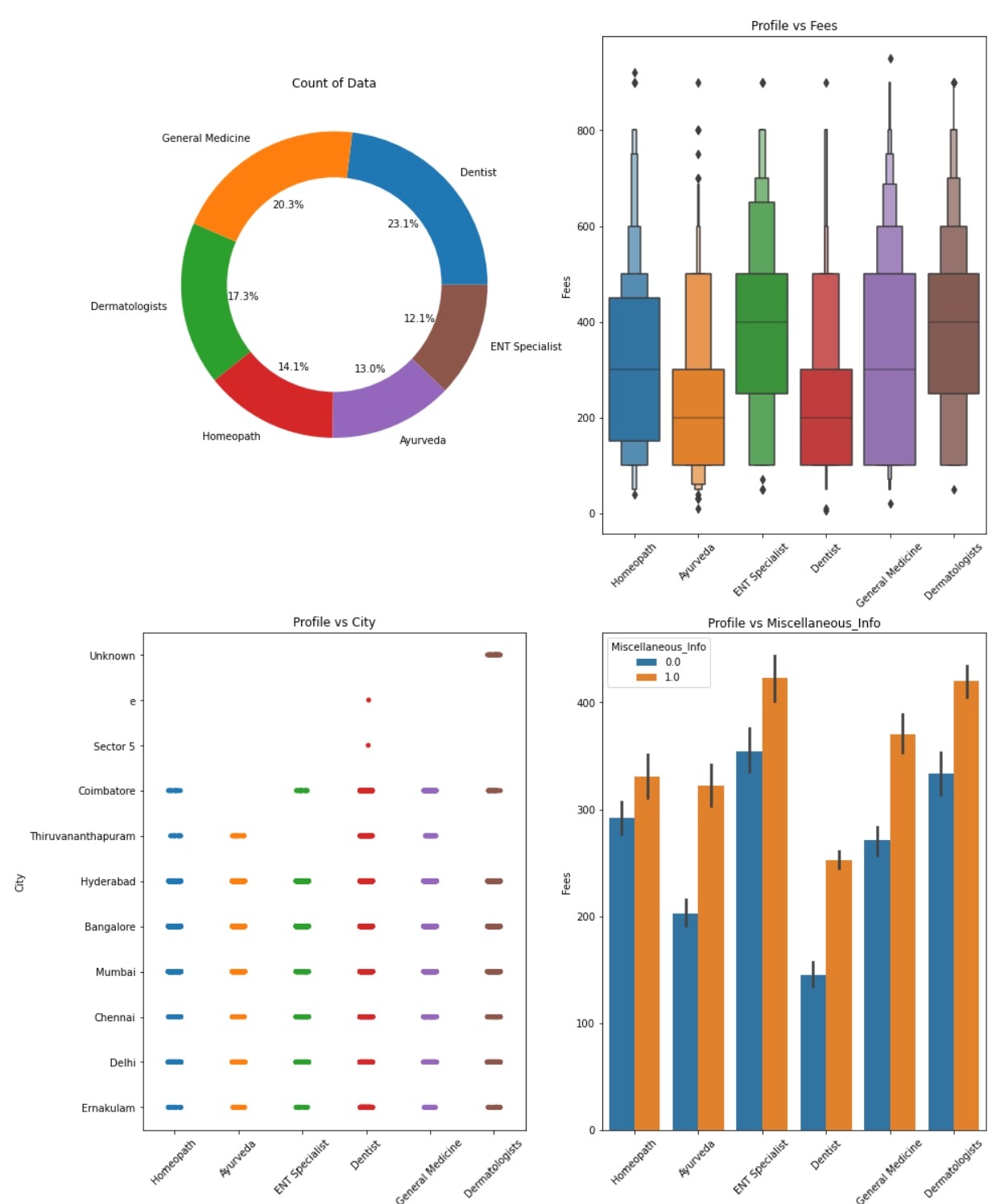




1. Most ratings lie in the range 80 to 100 while there are few ratings below 60. Data is highly skewed to the left.
2. As Rating increases, fees decreases.

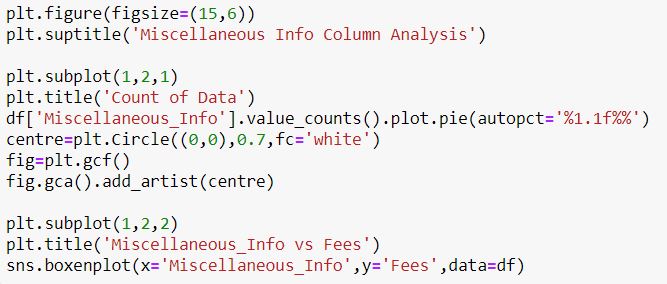
**Profile**

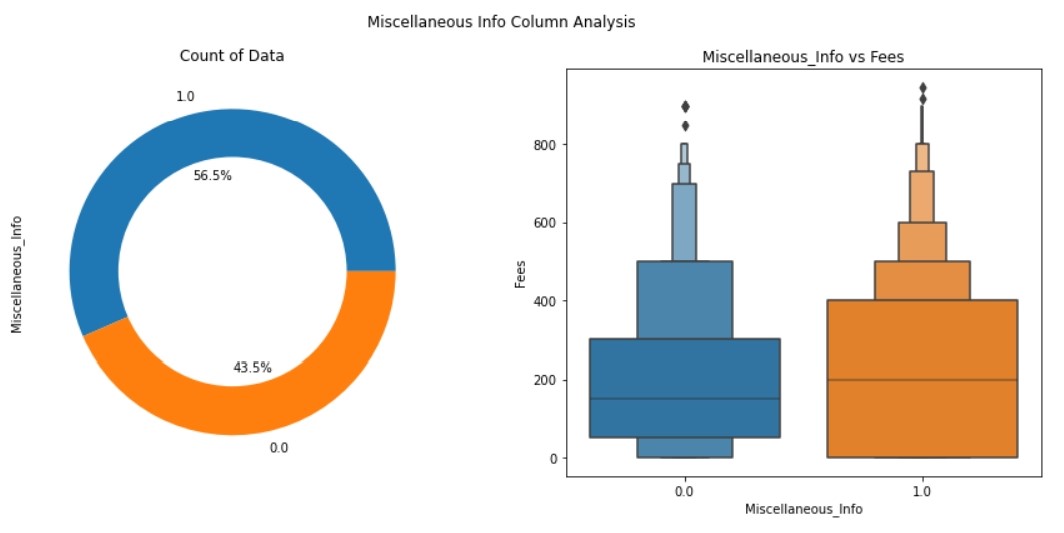




1. Most of the doctor’s profiles are of Dentist and General Medicine. ENT specialists are the rare ones as they are also referred to as super specialist.
2. ENT Specialists and Dermatologists charge the highest mean fees as compared to others while Ayurveda doctors charge the least.
3. Doctors only with Dermatologists profile have not provided their city whereas all the others have provided it. Dentists and General Medicine doctors belong from almost all the cities. Major cities such as Delhi, Chennai, Mumbai, and Hyderabad have all types of doctors available.
4. Doctors who have provided Miscellaneous info charge comparatively higher than the doctors who haven’t.

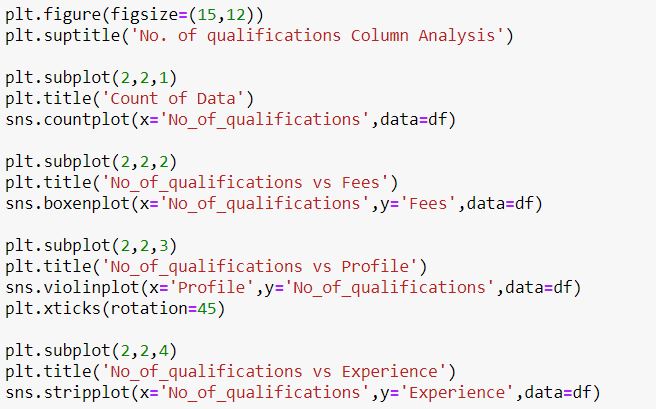
Miscellaneous Info

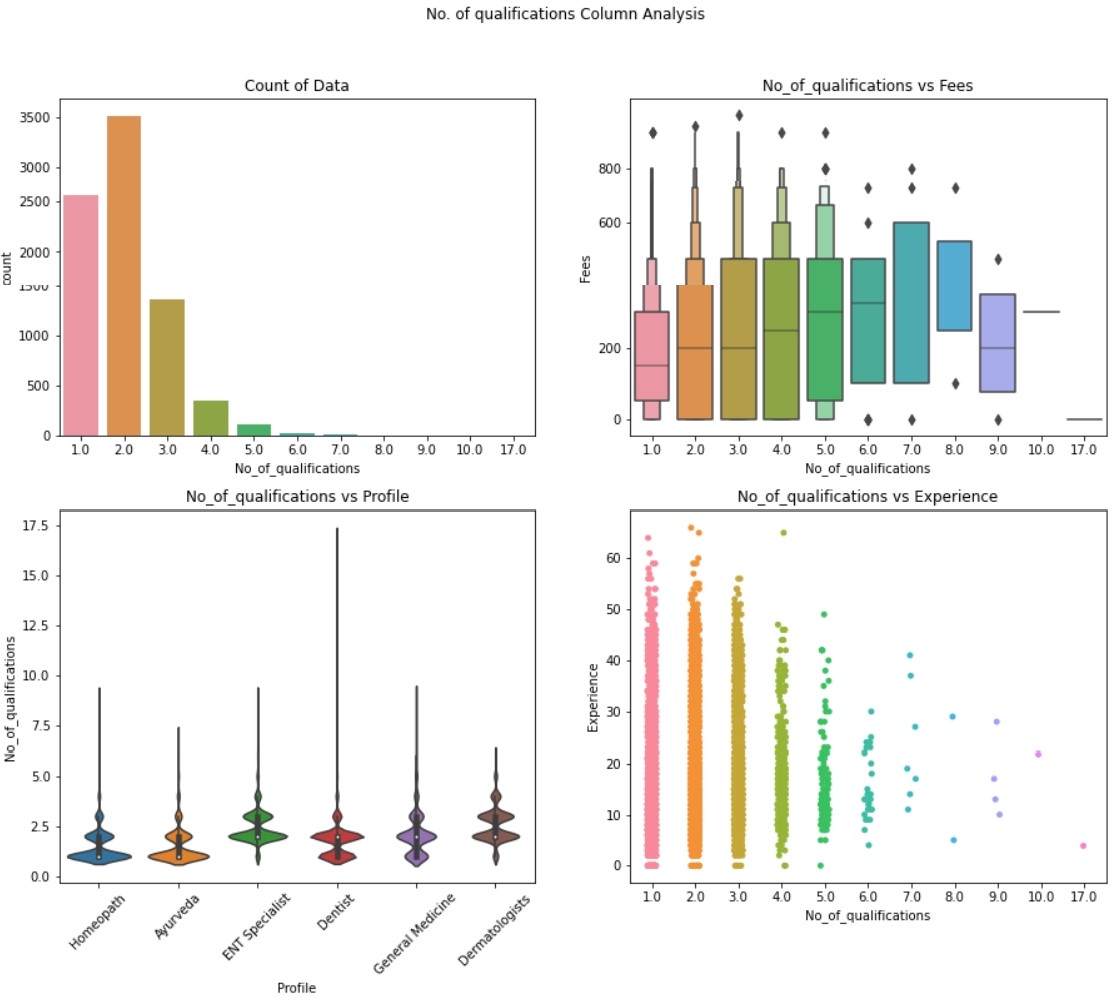




1. Around 56.5% of doctors have provided miscellaneous information.
2. Doctors who have provided miscellaneous info have mean fees higher than the doctors who haven’t.

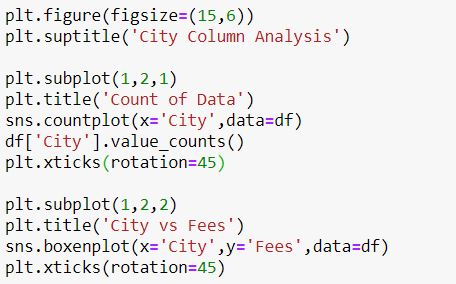
**No. of Qualifications**

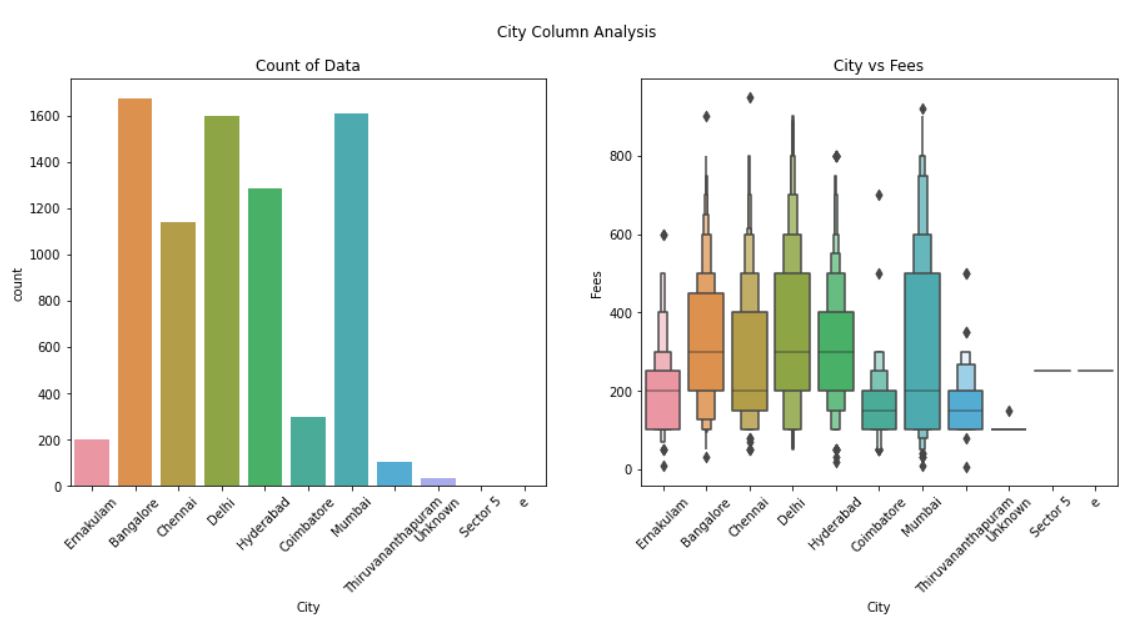




1. Most doctors have two qualifications, the number of doctors decreases as the no. of qualifications increases.
2. As the number of qualifications increases, fees of doctors also increases but there is a change after 8th no. of qualification.
3. Dermatologists and ENT Specialists have comparatively higher density for higher no. of qualifications, but dentists profile have outliers who have qualifications up to 17.
4. As the number of qualifications increases, experience in doctors decreases, which could be because doctors with a higher qualification earn more and tend to retire early or because they have to study more and get less time to practice.

**City**

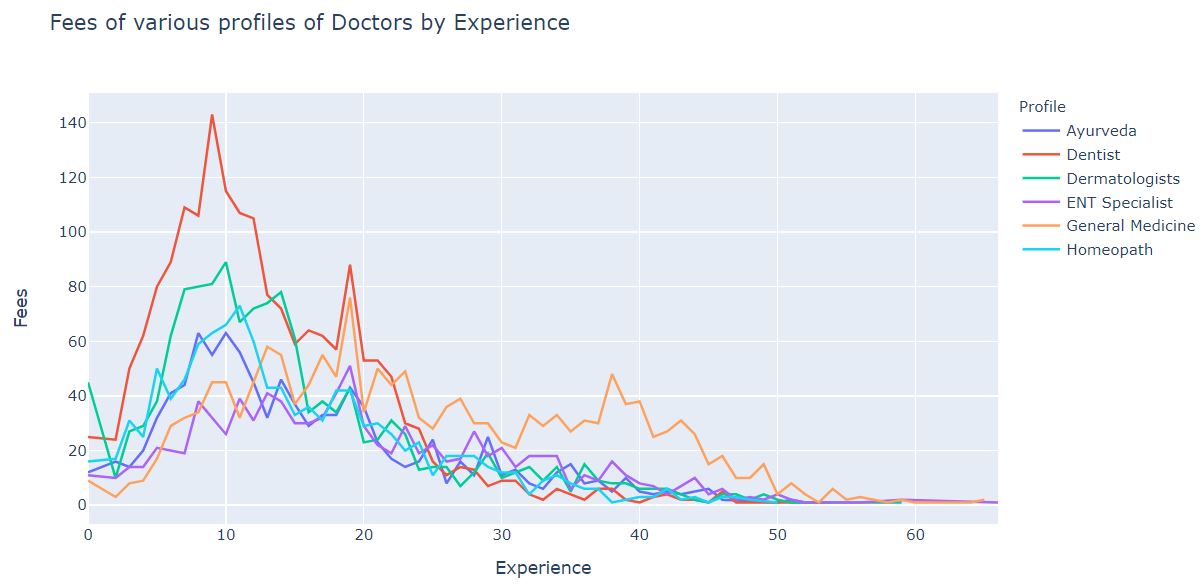




1. Metropolitan developed cities such as Bangalore, Hyderabad, Delhi have more doctors than other cities.
2. Doctors in these developed cities have higher mean fees as compared to other cities.

**Fees vs (Experience and Doctor’s Profile)**

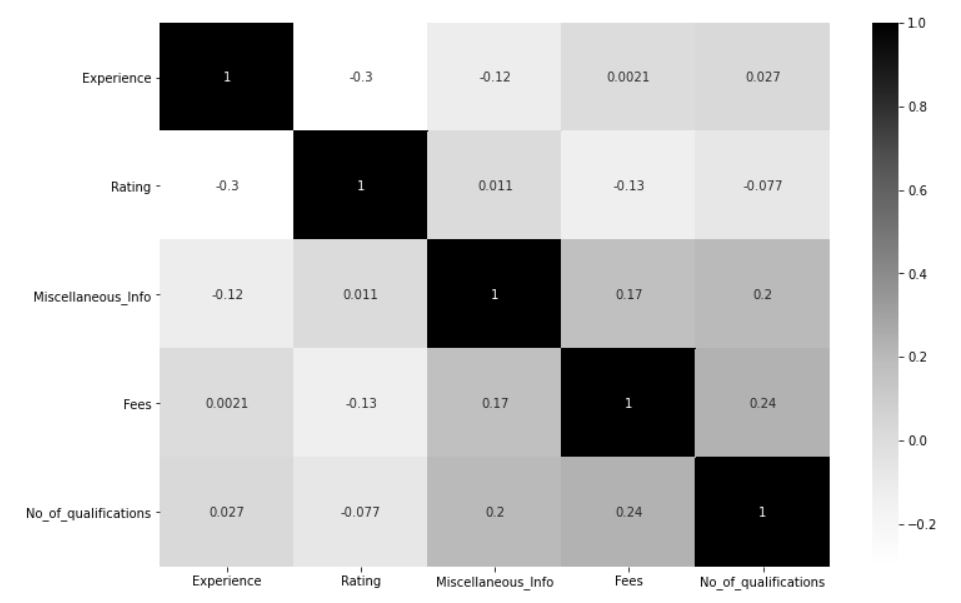




The highest peak in fees is found in dentist and Dermatologists when they have around 10 years of experience, fees decrease constantly after that with few local peaks. Charges of doctors decrease to a considerably low level after they have 40 years of experience except for general medicine who charge hefty fees even in their later years.

**Correlation between features**



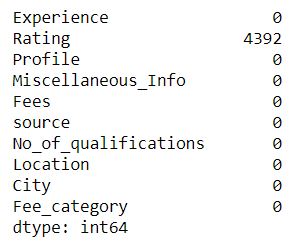


Fees show a positive correlation with No. of Qualifications and miscellaneous info. There is less correlation between independent features.

**Pre-processing Pipeline**

**Handling Null Values**





There are null values present in the rating columns as many of the doctors have not received any rating. We will fill the null values of the rating column with -999 to indicate our model that these ratings are totally different and, imputing -999 will also reduce the skewness of this column.



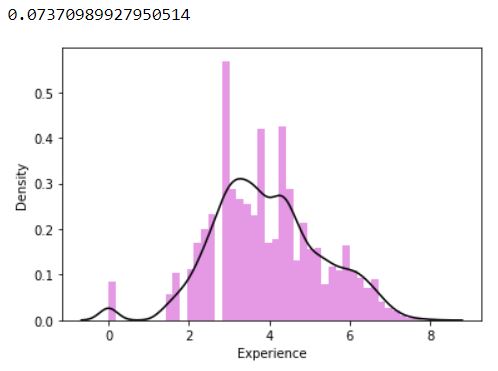


**Reducing skewness of Experience column**

We are going to remove skewness from this column using square root method







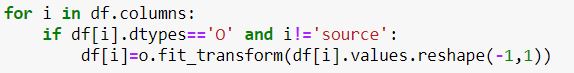
As we can see skewness is removed and is now very considerably low.

**Encoding**

We are going to use the Ordinal Encoder for encoding object type values of columns into float types.



Except for the source column, we are going to encode each column which is of Object datatype

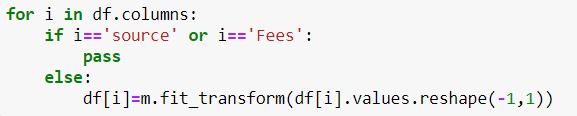


**Scaling**

We are going to use Min Max Scaler for scaling the column values of our dataset between 0 and 1 so that no column gets more or less weightage depending on their high or low values.



We scale all the columns except for our label column which is fees and the source column which serves as an identifier for which rows belong to the training set and which to test set.



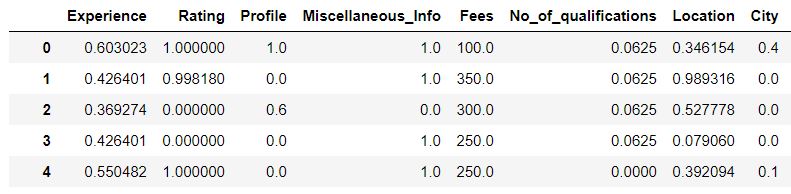
After cleaning, analyzing, and pre-processing our data we separate our train and test set as they were before with the help of the source column and by using the iloc function.



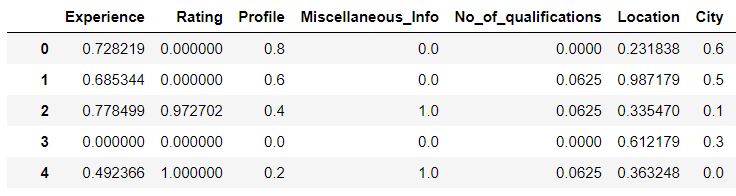
After separating the dataset we drop the source column from both the train and test set while the fees column only from the test set as we had created it only for concatenating the test set with the train set.



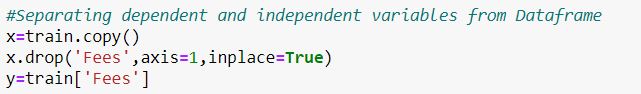
After all the processing this what our train set looks like.



The test set looks the same, only with the label column missing.

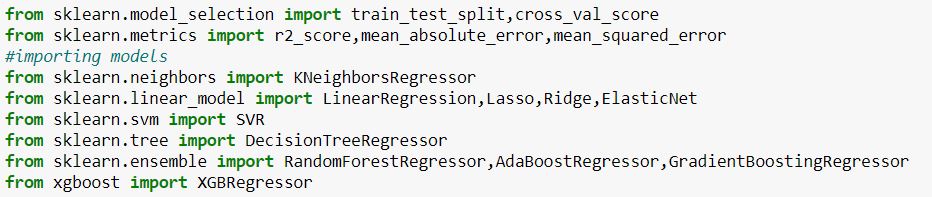


Further, we separate dependent and independent features from the training set for simplicity.

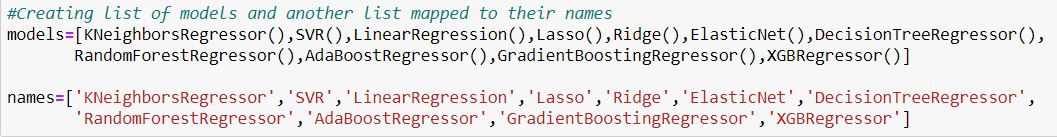


**Building Machine Learning Models**

We import the necessary modules and libraries.



We create a list of Regression models which we have imported for training and evaluating them one by one. We also make a list of names of these models for reference.



We split the training data into train and validation sets to train the models and then evaluate their performance on the validation set.

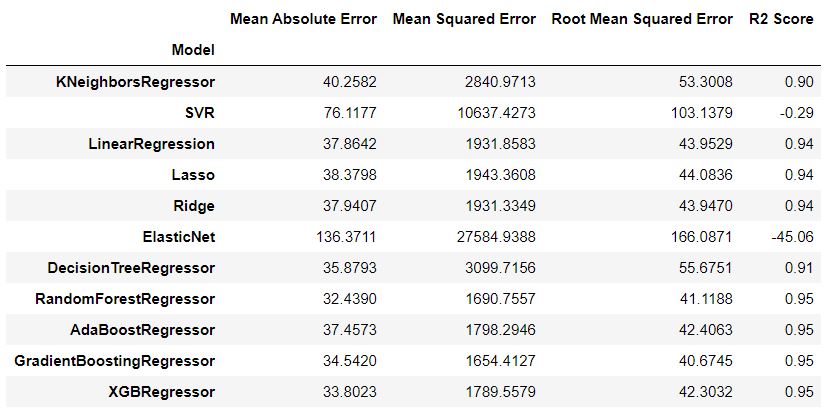


We create a function that takes the model list, name list of the models, training, and validation which we created above as input. Trains each model present in the model list, calculates various metrics such as mean absolute error, mean squared error, root mean squared error, and r2 score with the validation set. Appends each of these metrics in a list and finally clubs them up in a Data frame and returns it.



Upon calling the above function following data frame is returned.





Above data frame gives us the various performance metrics of each of the model and tells us how they perform in predicting consultancy fees of the doctors and on giving a close look it turns out that Random Forest Regressor does a better job than all the other models as it gives the least absolute, mean squared and root mean squared error while also giving a high r2 score.

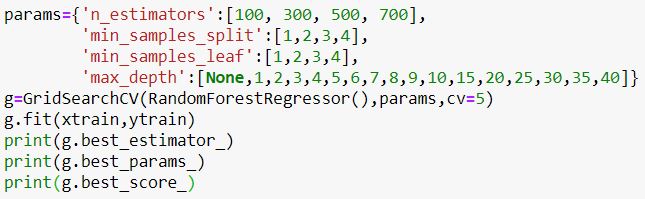
So, we further perform Hyperparameter tuning on it.

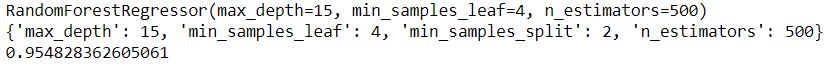
**Hyperparameter Tuning**

There are a lot of parameters available in the Random Forest Regressor. For now, we will try to tune it on the following four parameters.

* **n\_estimators**: The number of decision trees in the forest**.**
* **min\_samples\_split**: The minimum number of samples required to split an internal node
* **min\_samples\_leaf:** The minimum number of samples required to be at a leaf node. This may have the effect of smoothing the model, especially in regression.
* **max\_depth:** The maximum depth of the tree. It defines to what depth the decision tree needs to be expanded.

We use GridSearchCV to perform this process.

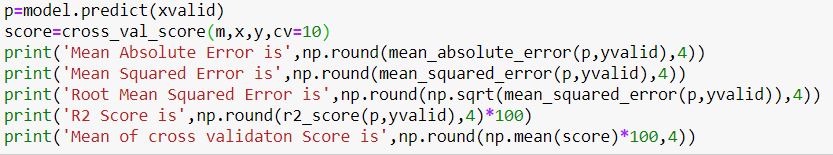


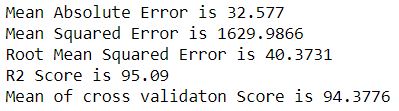


GridSearchCV returns the above parameters as the best one. So, we fit these parameters into our model.

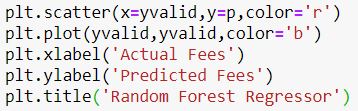


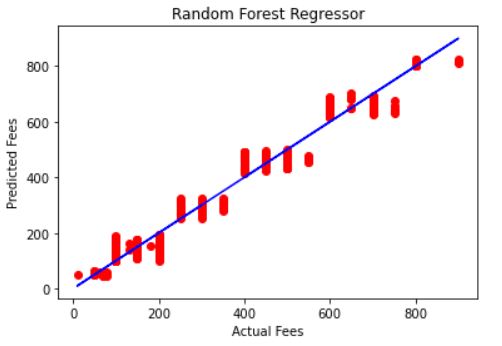
We use the final model that we have created to predict the validation set, perform K-Fold validation on it to see if our model is overfitting or not, and evaluate the performance metrics.





Our model gives a good score in performing the prediction for consultancy fees and also does not overfits the data. We can further visualize the performance of our model.





The above graph shows that the predicted fees are very close to the actual fees.

Predicting the test set

We can now predict the test set and convert the predictions into a data frame.

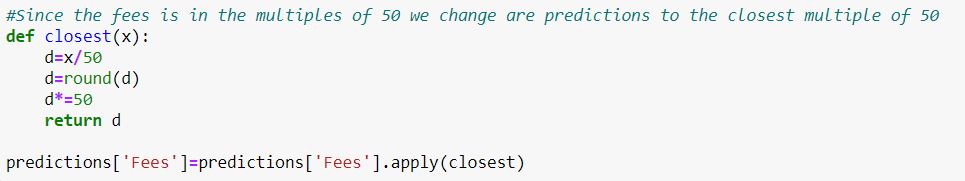




Our predicted fees look something like this.



But fees of doctors are in the multiples of 50 so we round off the fees into closes multiples of 50.



**Concluding Remarks**

**Reflection**

1. Consultancy fees of a doctor depend on the qualifications of the doctor and the area where they practice. If a doctor practices in an elite locality, his fee is going to be high.
2. Ayurveda and Homeopathy doctors have comparatively less fees than Allopathy and Specialists.
3. Another interesting thing was that doctors with more experience do not charge more. Doctors in their starting years of practice as well as after 30 years of experience charge less while they charge more in the middle years of their career.

**Acknowledgments**

[Student Support Team (datatrained.com)](https://support.datatrained.com/)

[scikit-learn: machine learning in Python — scikit-learn 0.24.2 documentation](https://scikit-learn.org/stable/index.html)

[Recently Active 'machine-learning' Questions - Stack Overflow](https://stackoverflow.com/questions/tagged/machine-learning)

For more details, please check out the source code on [Github](https://github.com/bitsplease98/Datascience-Practice-Projects/tree/main/Doctor's%20Consultancy%20fees%20prediction)