Part 1 - Regression

Performance, Evaluation and Improvements

Initial Approach

- Use **all columns - factorize string columns into classes using pandas** and get something working, even with **bad accuracy.**
- **Noticed a negative correlation coefficient.**

Refinements

- **Plot a correlation map.** Noticed that not all the columns were adding value to the model. So a lot of the columns were removed.
- **Remove** columns such as **homepage** and **outline** that were not useful for calculating **revenue.**

Further Refinements

- **I then plotted the Revenue distribution along with the frequency and the probability plot**. The target variable was **right skewed**. Linear models prefer **normally distributed data** so I transformed this variable to make it more normally distributed. I applied the log distribution.
- I also removed some values from budget and revenue that were really small, and all NA values in all columns.
- This dramatically improved my correlation coefficient.

Further Refinements

- Considering that I already had a good coefficient using just budget and runtime, I decided to play around with other columns and their lengths to see how they affect the correlation coefficient. After a few experiments, I noticed that using the length of cast, crew and genres columns added value to the model. So I decided to use them :).
- By this point, my Correlation Coefficient was 0.37 and I thought that was decent.

Problems you have faced in predicting

Problem Faced	**How I Resolved It**
All non numeric fields in the given data set are json strings	Reuse code from previous assignments and ‘ast’ python library to convert these into json objects
Most columns had data that was not directly useful for the model	I chose to only use some of the data from these columns
The data being skewed because of very small or very large values	Log transformation on revenue and discarding small budget and small revenue entries.
Values in the dataset were strings and wouldn’t comply with scikit	Factorising or applying a mapping function to use them more elegantly.

Part 2 - Classification

Performance, Evaluation and Improvements

Initial Approach

- Training the data with a **very large K value** and **using weights as uniform** using all fields. Again, this was done to get something working.
- The accuracy was almost 0.

Refinements

- **Similar to part1, a lot of columns were seen to be taking value away from the model. So I removed most of them.**
- **Converting K to 19 (k < sqrt(400)) and weights to distance.** Then keep decreasing K to get the best accuracy as learnt in lectures.

Further Refinements

- Decided to stick with K=17 after investigating all three evaluation properties for K=19,17,15,13,11.
- Was already getting an **average accuracy** without using data smartly.

Further Refinements

- This time, experimenting showed that using just budget and runtime produced the best accuracy, recall and precision.
- Some Feature Engineering seemed to improve the accuracy slightly, however, the avg_recall was being sacrificed so I chose to discard the Feature engineering.

Problems you have faced in predicting

Problem Faced	**How I Resolved It**
Same errors for json strings, arrays of objects with trivial data fields, worthless columns such as homepage and outline, non-compliant string values - Same errors as Part 1	Write functions that perform data cleansing and return a fresh dataset that has classified data with no strings and removes columns that are not needed
Errors in data classification such as labels being incorrect	Researching why these errors occur and using the right flags for setting the right labels