Capstone

mq2066

May 2024

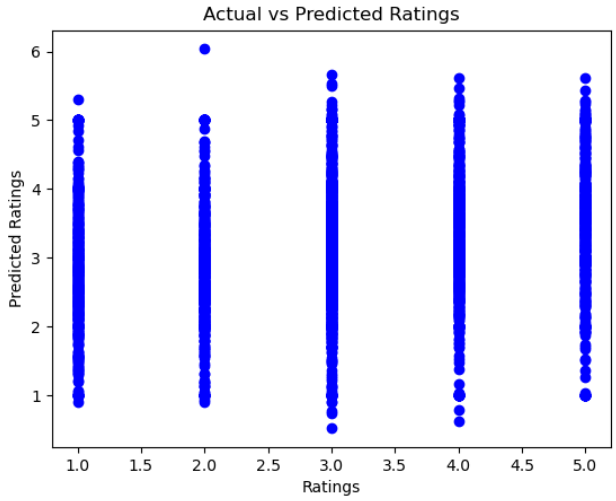
1 Regression

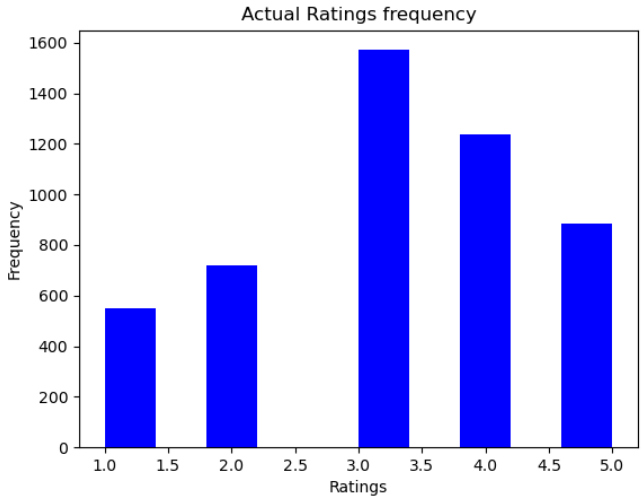
1. **Preparing Data:** First I created my dataframe by reading the csv file and the txt file to associate tuples of the form(movie title, rating, date) to a key, userid, in a dictionary, and then I convert that dictionary into the rows of a dataframe using Pandas. Next to create the train test split, I grouped the ratings by title, then I randomly sample one rating from each movie to be my test set, and drop this from the training set to prevent leakage. To construct the training set, I created a dataframe of movies and past ratings, so that each movie had the same amount of ratings(the average ratings per movie) and the system could be solved using least squares. I used numpy.pad to pad the rows with less than the designated ratings per movie with the mean rating for that movie and dropped the ratings that exceeded this amount. To account for possible collinearity between past ratings I remove instances where the same person may have rated the same movie more than once and only keep the first rating.

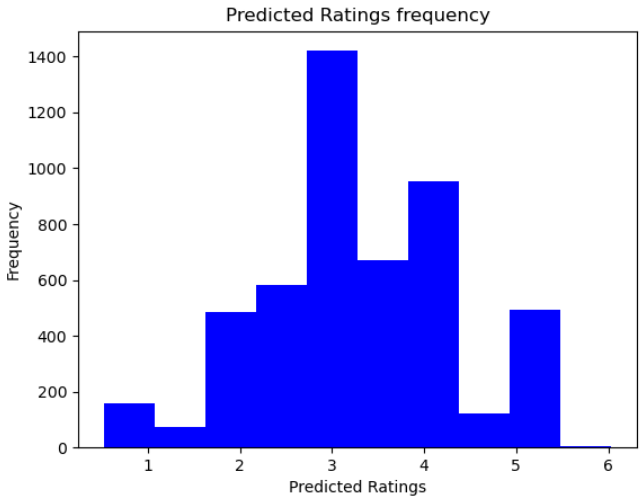
2. **Ordinary Least Squares Linear Regression:** Performing OLS on the train data results in the following features of the model. The F statistic measures how likely a null distribution is and this probability is 0, so our data is not randomly distributed. The adjusted R squared penalizes the high number of predictors but is still high at .896 suggesting a large amount of variance is explained by the model. However the log likelihood of the model is low and negative which suggests the model is not a good fit since Maximum likelihood estimation and least squares should yield the same coefficients assuming normally distributed error. The Jarque-Bera probability of 0 confirms that the model is a poor fit since the error is not normally distributed and the covariance type is nonrobust meaning it assumes normal distribution. The omnibus probability matches non normally distributed residuals as does kurtosis which suggests heavier tails than a Gaussian distribution should have. The condition number of the matrix shows it is an ill conditioned system and slight perturbations lead to a high error. Durbin Watson suggests the residuals are not autocorrelated.

0	1	2	3
Dep. Variable:	y	R-squared (uncentered):	0.957
Model:	OLS	Adj. R-squared (uncentered):	0.896
Method:	Least Squares	F-statistic:	15.73
Date:	Tue, 14 May 2024	Prob (F-statistic):	0.00
Time:	17:56:09	Log-Likelihood:	-5433.2
No. Observations:	4965	AIC:	1.668e+04
Df Residuals:	2058	BIC:	3 3.561e+04
Df Model:	2907		
Covariance Type:	nonrobust		

4	5	6	7
Omnibus:	323.795	Durbin-Watson:	1.961
Prob(Omnibus):	0	Jarque-Bera (JB):	1517.754
Skew:	-0.062	Prob(JB):	0
Kurtosis:	5.706	Cond. No.	1.01e+16

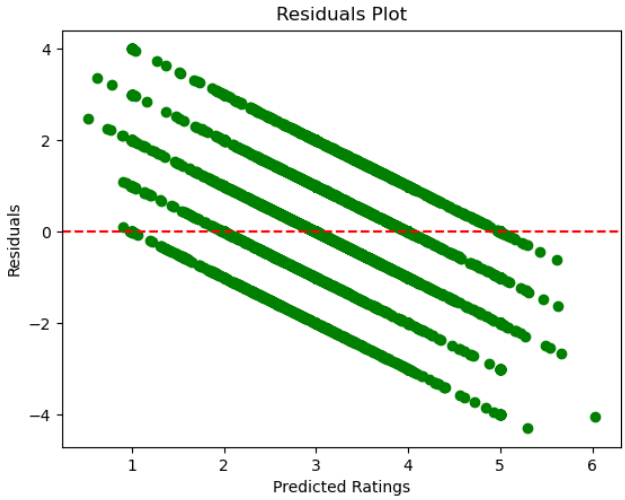






3. **RMSE:** The Root Mean Squared Error is 1.3572665955560554 which means the square root of average squared deviation from actual values is within 1.36 star of the actual. Because the same person may have rated multiple movies, there is some inherent collinearity in the data especially among similar movies, for example if a person favors a specific genre of movie and only watches and reviews those movies. I left these samples and assumed each user's rating is independent and dropped only cases where a person may have reviewed the same movie multiple times.

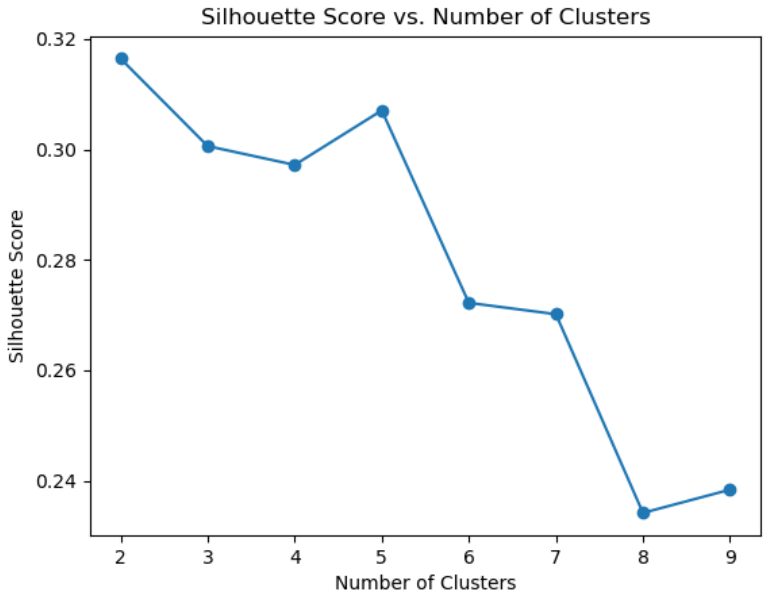
```
model = sm.OLS(y_train, X_train).fit()
predictions = model.predict(X_train)
mse = mean_squared_error(y_test, predictions)
print(np.sqrt(mse))
```



2 Classification

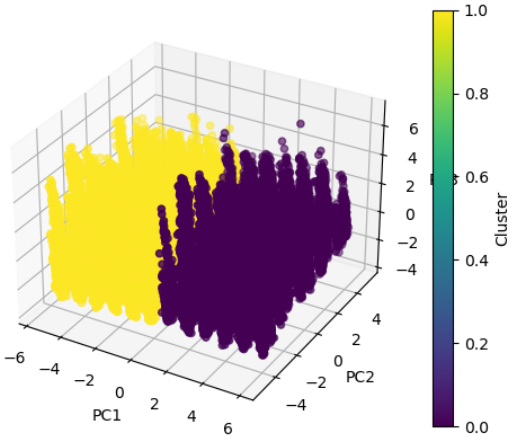
1. **Preparing Data:** To handle data I first replace missing values '?' with Nan and use pandas dataframe.dropna(). Then I use label encoder for the categorical variables, mode, key, and music genre. Mode is assigned binary dummy variables, while music genre and key are assigned integers values for each class. I then drop the columns, instance id and obtained date because these features are useless for classification and I drop the features artist name and track name for simplicity. To make the train test split, I first group by genre and randomly sample 500 songs from each, and this is my test set. I then drop these columns to create my training set, and split these into feature and target dataframes, then standardize the features.

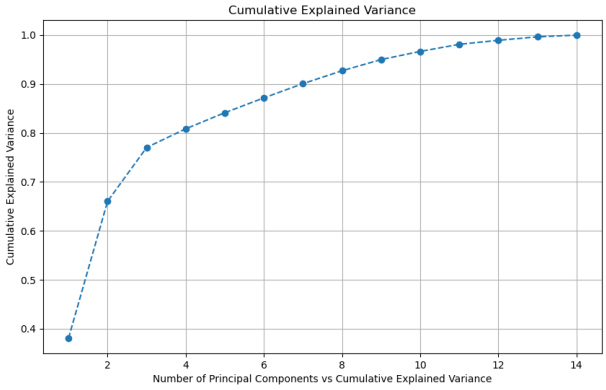
2. **Clustering:** I find the optimal number of clusters by calculating silhouette score against the number of clusters in the range (2,10). Silhouette score balances intercluster and intracluster distance to evaluate quality of clustering and the highest silhouette score at 2 clusters implies a binary classification problem when we know there are 10 genres. However the score of .31 when silhouette score ranges from -1 to 1 suggests that this clustering is still not optimal and the data may not be well separable.



3. **Dimension Reduction:** I standardize continuous columns using standard scaler to normalize since Principal Component Analysis is sensitive to scale. After doing principal component analysis I find the optimal number of principal components based on the Kaiser criterion where only eigenvectors with eigenvalues above 1 are selected, and I retain 5 principal components. Based on the cumulative explained variance chart, 5 principal components also explains about

.85 of the variance which is also enough based on the convention of accounting for at least 80% of the variance. To further prepare my data, I run principal component analysis on my training set and test set so my input will have 5 principal components instead of 13 features. Below the first three principal components are visualized with the optimal 2 clusters found by the k means algorithm.





4. **Feed Forward Neural Network:** I used a feedforward neural network with 32 neurons in each of the 3 hidden layers to classify my reduced dimension data. This consists of a linear layer where the input layer is multiplied by weights and the bias vector is added. It is then mapped to the hidden layer of dimension 32 then I use relu activation and introduce random dropout(probability=.5) after each activation. This introduces nonlinearity and regularization, respectively. Then the second linear layer maps to the hidden layer and is similarly followed by activation and dropout. Until the third linear layer which maps the hidden layer outputs to the softmax layer where probabilities are assigned to each input to classify them. After training my model I get an AUROC score of 0.8333373333333333, which implies decent model performance well above that of random guess. The Receiver Operating Characteristic Curve for each class is pictured below and plots true positives against false positives at all classification thresholds.