Predicting Song Popularity Using Machine Learning

Brad Shook and **Francisco Duarte Martinez**
{brshook,frduartemartinez}@davidson.edu
Davidson College
Davidson, NC 28035
U.S.A.

Abstract

For the music industry, predicting songs can be a profitable venture if executed properly, but its not an easy task. In order to achieve this, we attempted to create classification models that could accurately predict multi-class popularity labels for songs based on data provided by Spotify's API. As a result, we ended up creating a random forest, adaboost, and k-nearest neighbors classifiers as well as a neural network. In this attempt, the random forest performed slightly better than the other models with an accuracy score of 64.54% and an F1 score of 65.40%. While this is a low percentage point, both these metrics are nearly twice as high as the baseline which was 23.9%.

1 Introduction

The worldwide music industry is a multi-billion dollar industry. Recording studios contract many musicians with the hopes of them becoming cultural stars. If these recording studios and musicians were able to predict whether their songs would be widely popular, it would theoretically allow them to produce more popular songs. Using machine learning, we can test whether it is possible to predict the popularity of songs based on a variety of features.

For this research, the dataset was collected by Yamac Eren Ay using the Spotify API, which has a number of song features for every song (Ay 2021). These features include acousticness, tempo, liveness, and others (see Appendix for full list of features and descriptions). The popularity feature was used as our label for each song. Popularity scores ranged from 0 to 100. These popularity scores are calculated by an algorithm and are based on the total number of plays the track has had and how recent those plays are (Spotify 2021).

Most machine learning research surrounding songs has been focused on classifying songs into musical genres. Due to this, there is little to no public research on predicting song popularity available. Internally, it appears that record labels have attempted to predict the popularity of songs based on some features, but it proved to be a difficult challenge in the past (Pachet and Roy 2008).

The rest of this paper will show the various data preprocessing techniques that we employed, the process of tuning model hyperparameters, and the evaluation of several different classification models to determine which model classifies song popularity most effectively.

2 Data Preparation

The dataset contained 174,389 songs with 19 features each, including popularity. We chose to only use the numerical features and excluded the features that were strings. This left us with 9 features. As mentioned earlier, the popularity scores were used as labels. Since the popularity scores range from 0 to 100, we had the choice of treating this as a regression problem or a classification problem. We chose to treat it as a classification problem. Since there are 101 possible popularity scores, we chose to split them into a smaller number of classes. However, the boundaries of these splits had to be made in a way in which the number of songs in each split were similar in counts to avoid data imbalance. Thus, we analyzed the distribution of the popularity scores among the songs to find the best boundaries.

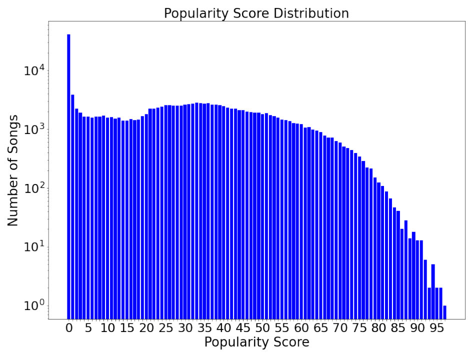

Figure 1: Bar graph showing the number of samples for each popularity score.

In Figure 1, we see that there are less songs with high popularity than those with low popularity. Additionally, the popularity score with the highest number of songs

Label	Boundary	Number of Songs	Proportion
0	[0]	40,905	23.5%
1	(0,22]	39,180	22.5%
2	(22,38]	41,770	23.9%
3	(38,50]	25,634	14.7%
4	(50,100]	26,900	15.4%

Table 1: Boundaries and number of songs for each class.

was 0. Because of this, the popularity score, 0, had its own class which included 40,905 songs. Then, we created four more classes. The boundaries and number of songs in each class are shown in Table 1. These five classes contained data that was balanced enough as to not affect the classification performance of our models. Note that the popularity score class of 2 had the highest proportion of songs so it is our baseline classifier.

After creating our classes, we randomly shuffled the data and split it into training (60%), validation (20%), and test (20%) sets. Then, the two features that had large ranges were scaled using a min-max scaler. These two features were loudness and tempo. Each was scaled to be between 0 and 1. This was the last data preparation technique that we performed on the data.

3 Experiments

For our experiments, we created four distinct models: a random forest classifier (RFC), an adaptive boosting classifier (ADAC), a k-nearest neighbors classifier (kNNC), and a neural network (NN).

The following experiments focused on tuning the hyperparameters of each of these models and comparing the efficacy of each model.

Experiment 1

In the first experiment, we tuned the hyperparameters of the RFC, ADAC, and kNNC using grid searches. These grid searches searched over different hyperparameters for each model (see Section 8). Each grid search used weighted F1 scores to compare models. Additionally, the grid searches used 5-fold cross-validation to find an average measure of performance in each model that was fitted. The mean weighted (MW) F1 scores were measured on the cross-validation folds.

The NN was tuned in a different way. We manually created multiple models with different structures and hyperparameters (see Section 8 for more details). Our goal was to create a model that maximized F1 scores and minimized the structural complexity of the neural network. This encompassed tuning the batch size, learning rate, number and types of hidden layers, activation functions, and number of epochs.

Experiment 2

For the second experiment, we compared the F1 scores and accuracies of each tuned model and the individual F1 scores for the popularity score classes. These F1 scores and accuracies were evaluated on the test set. By calculating the F1 scores, we were able to determine which model performed the best and which classes were most classifiable. Lastly, we analyzed whether the models were overfit or underfit by comparing the F1 score differences between the train and test sets.

4 Results

Results of Experiment 1

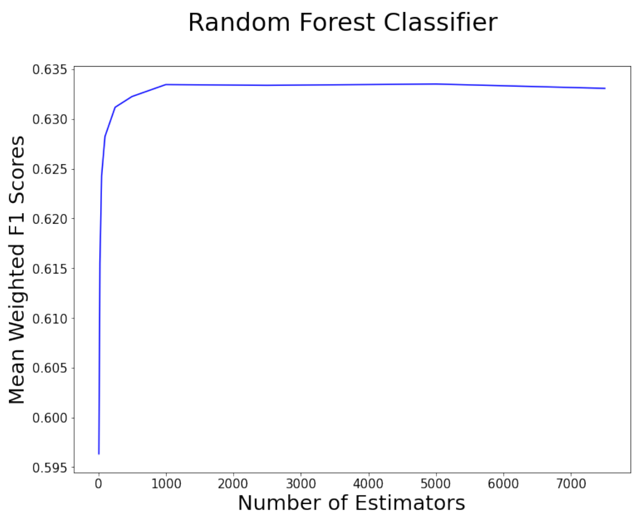

Figure 2: Line chart showing how different numbers of estimators affect the MW F1 scores. F1 scores are from a 5-fold cross-validation grid search.

For the RFC, our grid search found that 5,000 estimators was the value that yielded the best MW F1 scores. In Figure 2, we see that the scores peaked at 5,000 estimators and then plateaued as the number of estimators increased. While 5,000 estimators provided the best performing model, 1,000 and 2,500 estimators provided similar results while creating less resource intensive models.

Then, the grid search for the ADAC determined that a learning rate of 0.01 and 20,000 estimators resulted in the highest MW F1 scores. A heatmap displaying the full results from this grid search is shown in Figure 3. For ADAC, the biggest difference in MW F1 score is based on the combination of estimators and learning rate. In this heatmap, we see that models with a learning rate of 0.001 were the worst performing while the rest of the learning rates performed within 0.01 of each other depending on the amount of estimators. For a simpler but well performing model, a learning rate of 1.0 and 100 estimators yielded a MW F1 score just 0.0057 worst than the best performing model so this might be ideal for those

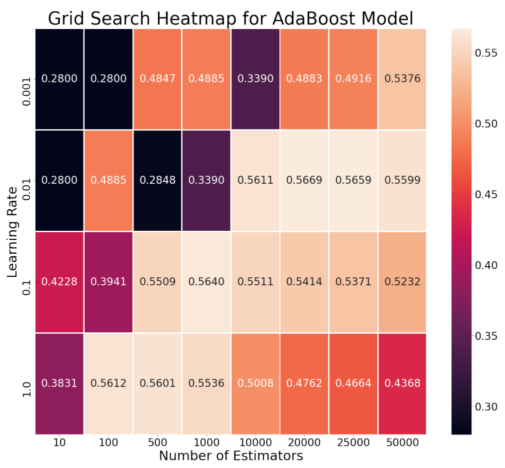

Figure 3: Heat map showing showing the MW F1 scores of each model created in the grid search. Each MW F1 score is from 5-fold cross-validation.

with limited resources.

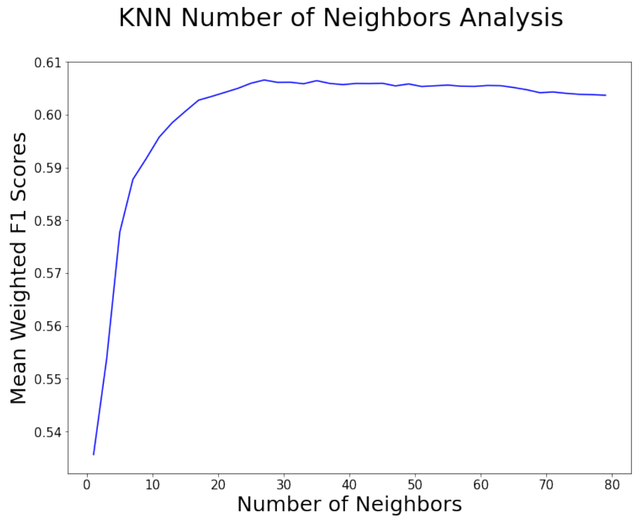

Figure 4: Line chart showing how different numbers of neighbors affect the MW F1 scores. F1 scores are from a 5-fold cross-validation grid search.

The last grid search for the kNNC found that 27 was the optimal number of neighbors. As the number of neighbors increased, there was slight decrease in MW F1 scores as shown in Figure 4

For the NN, we determined that a moderately deep neural network yielded the highest F1 scores. Specifically, our best performing NN had a 9 node input layer with 6 hidden layers, which each had a dense layer followed by a batch normalization layer followed by a ReLU activation layer (see Section 8 for model structure). The number of nodes in the dense layers began at 512 and decreased by a factor of 2 in the following dense layers until the last dense layer had 16 nodes. Then, the output layer consisted of 5 nodes and used the softmax activation function. The NN used the categorical cross entropy loss function with the Adam optimizer and a learning rate of 1e-4. These hyperparameters maximized the average F1 scores on the test set while minimizing the number of weights in the model.

Results of Experiment 2

Model	**Accuracy**	**F1 Score**
ADA Classifier	58.69%	59.97%
Random Forest	64.53%	65.40%
kNNC	61.62%	62.56%
NN	60.73%	62.02%

Table 2: Accuracy and F1 Scores on the test set.

After fitting each model with their tuned hyperparameters on the training set, we found that the RFC model produced the best F1 and accuracy scores. To see the full results, refer to Table 2. Overall, all of our models outperformed the baseline accuracy score of 23.9% by at least 34%. Both the kNNC and NN models had similar accuracies and F1 scores. However, the RFC outperformed the second best model, kNNC, by nearly 3 percentage points in both accuracy and F1 scores.

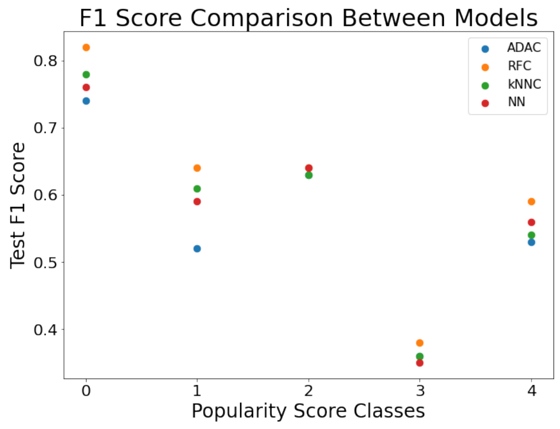

Figure 5: Scatter plot comparing the test F1 scores for each model when classifying each popularity score class.

In Figure 5, we can see that the most classifiable popularity score class was class 0. Contrarily, the class which the models were worst at classifying was class 3. At first glance, this would seem to be caused by class 3 having fewer samples. However, class 4 has similar F1 scores compared to class 1 which contains more samples, so, the difference in number of samples is not affecting the performance of class 4. This implies that class 3 is simply harder to classify than the rest of the classes. In terms of the individual models, the RFC yielded the highest F1 scores for each class. Each model classified songs from class 2 with similar F1 scores, while class 1 saw the most variety in F1 scores among the models.

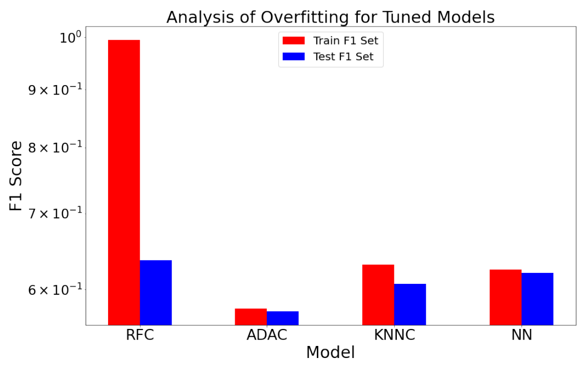

Figure 6: Comparison of F1 scores on test and train sets for each model.

As we saw in Figure 5, F1 scores on the test set for each model vary in efficacy. In Figure 6, we see that all four models were overfit by varying amounts in terms of F1 scores. The RFC was the most overfit model with the train F1 score being 0.358 greater than the test F1 score. The other three models were much less over fit. The reason that the RFC is so overfit is due to the nature of random forests attempting to create overfit models.

5 Broader Impacts

In the multi-billion dollar music industry, using machine learning approaches similar to those presented in this paper can have huge monetary consequences. While our paper focused on a set of discrete labels, real world uses will be vastly different as there are 101 labels in the original dataset and some might decide to focus on the binary accuracy of predicting whether a song is considered popular or not. Similarly, analysis created by music studios contains more information than that used in this paper. For example, music trends change every couple of years but this paper considered all music to be similar. Additionally, creating a perfectly accurate model may have unintended consequences. For instance, if an artist creates a song that defies the notion of a popular song and this model, or a similar one, is used to gauge the potential popularity of their song, the predicted popularity will certainly be lower. Similarly, if an artist is able to make music that is popular every time, the model will eventually become inaccurate as listeners become accustomed to previous songs. As a result, artists may decide to create music that will all sound the same, killing all creativity in the music industry. Finally, models like this may provide useful information on certain song features used by artists. These models can be expanded to show the impact of individual features on the popularity of a song allowing artists to focus their music on those characteristics. The previous implications still apply but their negative impacts may be minimized by not catering to the predictive model as a whole.

6 Conclusions

In this paper we attempted to create classification models that accurately predicted Spotify songs' popularity ranging from 0 to 4 based on Spotify API features. In our attempt, we created RFC, ADAC, kNN and Neural Network models. The models had accuracies and F1 scores in the low 60 and high 50 percentiles which were much greater than our baseline classifier of 23.9%. Our best model, the RFC, had an accuracy of 64.53% and an F1 score of 65.40%.

Further Work

While the ensemble methods that we used are some of the most successful machine learning classifiers, the kNN classifier also performed well on this task. Consequently, it may be beneficial to further explore the benefits of a kNN based classifier that uses ensemble methods (Gul et al. 2018) or cluster based classification (Jurek et al. 2012)(AbedAllah and Shimshoni 2012). Similarly, neural networks are also very strong at classifier tasks, so creating drastically different networks than the ones presented in this paper may provide more accurate results. Lastly, it may be beneficial to further explore the impact of other features not used for this approach. For example, implementing transformers for the name of an artist may be useful because an artist's name has a large impact on popularity in the music industry. Different combinations of the aforementioned strategies may result in more effective models.

7 Contributions

B.S. and F.M. both worked on data normalization and exploratory data analysis. F.M. built and tuned random forest, adaptive boosting, and kNN classifiers. B.S. built neural networks and tuned them. Both B.S. and F.M. created figures for the paper. B.S. wrote the introduction, data preparation, experiments, and part of results section. F.M. wrote part of the results section, as well as, the abstract, broader impacts, conclusion, and further work sections. Both B.S. and F.M. edited and reviewed the final paper.

References

AbedAllah, L., and Shimshoni, I. 2012. k nearest neighbor using ensemble clustering. In Cuzzocrea, A., and Dayal, U., eds., *Data Warehousing and Knowledge Discovery*, 265–278. Berlin, Heidelberg: Springer Berlin Heidelberg.

Ay, Y. E. 2021. Spotify dataset 1922-2021, 600k tracks. `https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks`.

Gul, A.; Perperoglou, A.; Khan, Z.; Mahmoud, O.; Miftahuddin, M.; Adler, W.; and Lausen, B. 2018. Ensemble of a subset of knn classifiers. *Advances in Data Analysis and Classification* 12(4):827–840.

Jurek, A.; Bi, Y.; Wu, S.; and Nugent, C. 2012. A cluster-based classifier ensemble as an alternative to the nearest neighbor ensemble. In *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*, volume 1, 1100–1105.

Pachet, F., and Roy, P. 2008. Hit song science is not yet a science. 355–360.

Spotify. 2021. Spotify web api documentation. https://developer.spotify.com/documentation/web-api/reference/#reference-index.

8 Appendix

Features Used

The following is all of the features used to create our predictive models and the descriptions provided by the Spotify API(Spotify 2021)

Feature	Description
Acousticness	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
Danceability	Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.
Energy	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy.%
Liveness	Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live
Loudness	The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude).
Speechinesss	Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value.
Tempo	The overall estimated tempo of a track in beats per minute (BPM).
Valence	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).
Year	The year the track was released.
Popularity	The popularity of a track is a value between 0 and 100, with 100 being the most popular. The popularity is calculated by algorithm and is based, in the most part, on the total number of plays the track has had and how recent those plays are.

Grid Search

The following shows the values searched over in the grid searches mentioned in Section 3.

- RFC
  - Number of Estimators:10, 25, 50, 100, 250, 500, 1000, 2500, 5000, 7500
- ADAC
  - Number of Estimators: 10, 100, 500, 1000, 10000, 20000, 25000, 50000
  - Learning Rate: 0.001, 0.01, 0.1, 1.0
- KNN
  - Nearest Neighbors: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79
- NN
  - All models have an input layer of size 9 with a ReLu activation function and output layer of 5 with a softmax function. Additionally, batch normalization is applied to every layer before the activation function and all hidden layers used ReLu as their activation functions. See Table 3 for the structures and results of the tested neural networks.

Hidden Layers	Loss Function	Epochs	Learning Rate	Weighted Average F1 Scores
16	Adam	200	1e-4	0.55
32	Adam	200	1e-4	0.57
64	Adam	200	1e-4	0.56
32 → 16	Adam	200	1e-4	0.59
64 → 32 → 16	Adam	200	1e-4	0.59
128 → 64 → 32 → 16	Adam	200	1e-4	0.59
256 → 128 → 64 → 32 → 16	Adam	200	1e-4	0.60
512 → 256 → 128 → 64 → 32 → 16	Adam	200	1e-4	0.61
1028 → 512 → 256 → 128 → 64 → 32 → 16	Adam	200	1e-4	0.60
16 → 32 → 64 → 128 → 256 → 512 → 128 → 64 → 32 → 16	Adam	200	1e-4	0.61
16 → 32 → 64 → 128 → 256 → 512 → 128 → 64 → 32 → 16	Adam	400	1e-4	0.61
16 → 32 → 64 → 128 → 256 → 512 → 128 → 64 → 32 → 16	Adam	200	1e-2	0.61
16 → 32 → 64 → 128 → 256 → 512 → 1028 → 128 → 64 → 32 → 16	Adam	200	1e-4	0.61
16 → 32 → 64 → 32 → 16	RMSProp	200	1e-6	0.57
16 → 32 → 64 → 32 → 16	RMSProp	400	1e-6	0.57
16 → 32 → 64 → 32 → 16	RMSProp	600	1e-6	0.58
16 → 32 → 64 → 32 → 16	RMSProp	200	1e-4	0.59
16 → 32 → 64 → 32 → 16	Adam	200	1e-4	0.59
16 → 32 → 64 → 32 → 16	Adam	400	1e-5	0.58
16 → 32 → 64 → 32 → 16	Adam	600	1e-5	0.59
16 → 32 → 64 → 32 → 16	Adam	100	1e-5	0.57

Table 3: Neural networks created

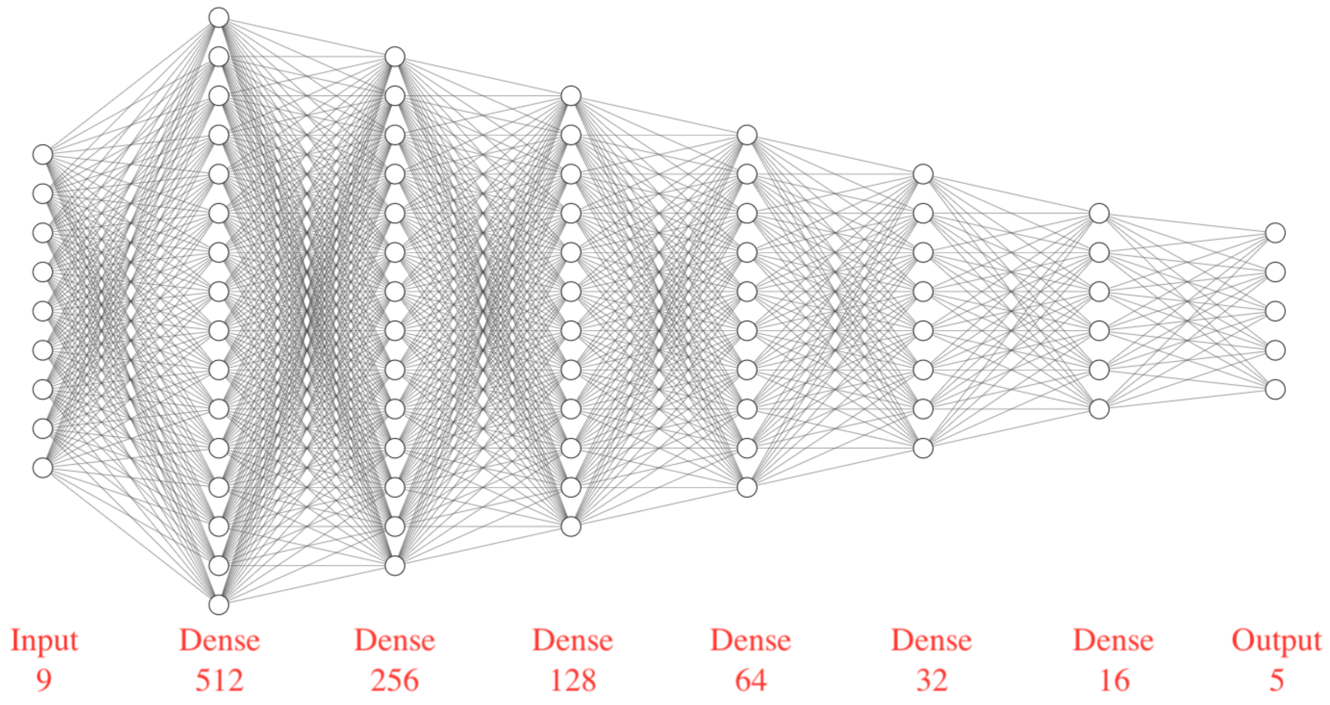

Figure 7: Structure of the best-performing neural network model. Note each dense layer was followed by a batch normalization layer then an activation layer.