Predicting Assigned Grade Distributions from Class Attributes

I began with grade data from the University of Wisconsin at Madison for classes taught between the spring of 2007 and the spring of 2018. For each class, I also had access to a number of other attributes, such as schedule, classroom, type of class, instructor, and course number. My goal was to predict the grade distribution for a possible future class using only those attributes that could reasonably be expected to be known when students were registering for classes for the following semester.

From the beginning, it was clear that some of the features that I had could not be reliably known at the time of registration. For example, instructors are often only assigned shortly before classes start. In addition, because numbers are not fixed until several days or weeks into a semester, class enrollments cannot be assumed to be knowable at registration. Other features, such as subject, were categorical and contained so many different values as to be nearly useless.

After consideration, I selected the following features to use in my models:

- `section_type`: lecture, discussion, seminar, laboratory, or fieldwork,
- `start_time`: in minutes after midnight,
- `class_length`: in minutes,
- `term_code`: a four digit code indicating the year and semester a particular class was taught,
- `class_level`: the hundreds digit of the course number,
- `class_meetings`: the number scheduled class meetings per week,
- `median_enrollment`: the median enrollment of all classes assigned to the same classroom, and
- `number_of_sections`: the number of sections of the same course offered in the same semester

There were also a number of features that I chose to discard: `end_time`, `mon`, `tues`, `wed`, `thurs`, `fri`, `sat`, `sun`, `subject_code`, `max_enrollment`, `min_enrollment`, `third_quartile`, and `section_number.`

I next fit a variety of types of models to my features with the aim of predicting the percentage of A grades, AB grades, B grades, etc. assigned for a given class. For each type of model, I tuned parameters by fitting on the first 16 semesters of data and then validating on the subsequent 3 semesters. Once the best parameter combination had been found, I fit the model using the first 19 semesters of data, leaving the last 3 semesters as testing data.

I created pipelines for each of the following models (with the parameters found above) and fitted them to the first 19 semesters worth of data:

- Standard linear regression,
- Two different K nearest neighbors models:
  - A KNN regressor model with K = 16 and
  - A KNN regressor model with K = 7,
- Random forest regression,
- AdaBoost regression,
- Support vector regression,
- ElasticNet, and, finally

- A multilayer perceptron with three hidden layers.

I then evaluated the performances of the fitted models on my test set using the following metrics:
- Mean squared error (MSE),
- Mean absolute error (MAE),
- Median absolute error (Med AE), and
- The coefficient of determination (R2).

I obtained the following results:

	MSE	**MAE**	**Med AE**	**R2**
Support Vector Regressor	0.0175	0.0749	0.0254	0.0279
Linear Regression (base)	0.0170	0.0794	0.0370	0.1166
ElasticNet	0.0168	0.0788	0.0361	0.1210
AdaBoost	0.0164	0.0783	0.0372	0.1226
K nearest neighbors 7	0.0146	0.0695	0.0290	0.1427
K nearest neighbors 16	0.0141	0.0695	0.0300	0.1871
Multilayer Perceptron	0.0138	0.0688	0.0296	0.2193
Random Forest Regressor	0.0135	0.0694	0.0307	0.2281

We see here that none of these models is spectacular, which suggests that there are a lot of other factors that influence grade distributions. Of the models produced based on the data that we have, the multilayer perceptron and random forest regressor perform similarly, though the random forest regressor is perhaps marginally better.