Predicting Professor Ratings

Arik Gershman

Presentation Outline

01 About the project

02 Data

03 Models

04 Evaluation and Conclusion

01

About the project

About the project

PlanetTerp

Website where UMD students rate professors using stars and written reviews

The Problem

To predict professor ratings using information other than actual ratings

Purpose

Can be used by UMD to evaluate potential hires

02

Data

Data

PlanetTerp API

Used the Python Requests library and JSON parsing to get the data from PlanetTerp's API

Limits

PlanetTerp limits 100 items per API call, so Implemented a loop to offset API calls and retrieve data in batches of 100 items

Professors and Courses Endpoints

Professors: Name, Slug, Type, Courses, Average Rating, Reviews
Courses: Average GPA, Professors, Department, Course Number, Name, Title, Recent, Gen-Eds

Data

Sentiment Analysis

Target Column

Feature	Name	Negative Reviews	Positive Reviews	Number of Reviews	Average Expected Grade	Average Courses GPA	Number of courses	Average Rating
Description	The professor's name	Number of negative reviews	Number of positive reviews	Negative + positive reviews	Average expected grade for all students reviewing this professor	Average GPA of all the courses this professor teaches	Number of courses this professor teaches	The actual rating of this professor
Example	Maksym Morawski	48	36	84	2.791626	2.884761	15	2.8079

Notes

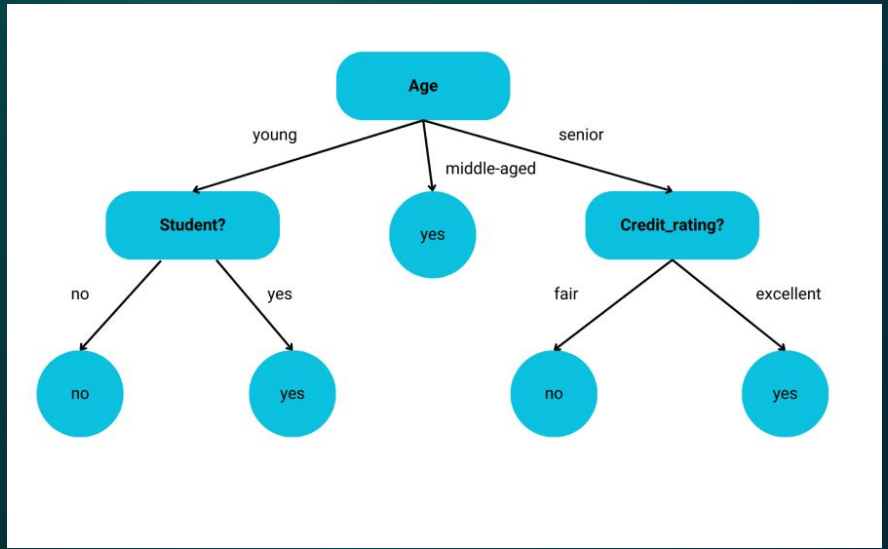
- 4177 total professors after removing ones with no reviews
- Used transformers.pipeline for sentiment analysis
- Average courses GPA is -1 if the professor doesn't teach any courses

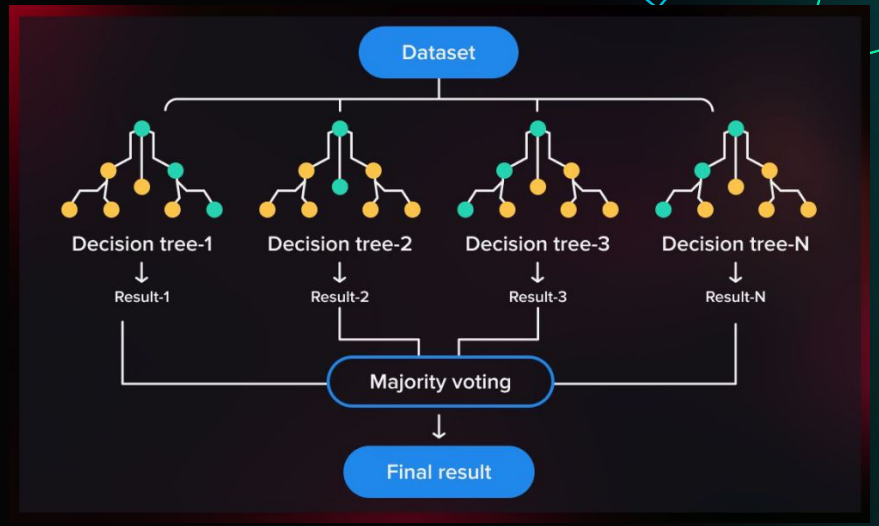
03

Models

Decision Trees

- Tree structure that splits data based on features
- Simple and easily interpretable
- Prone to overfitting



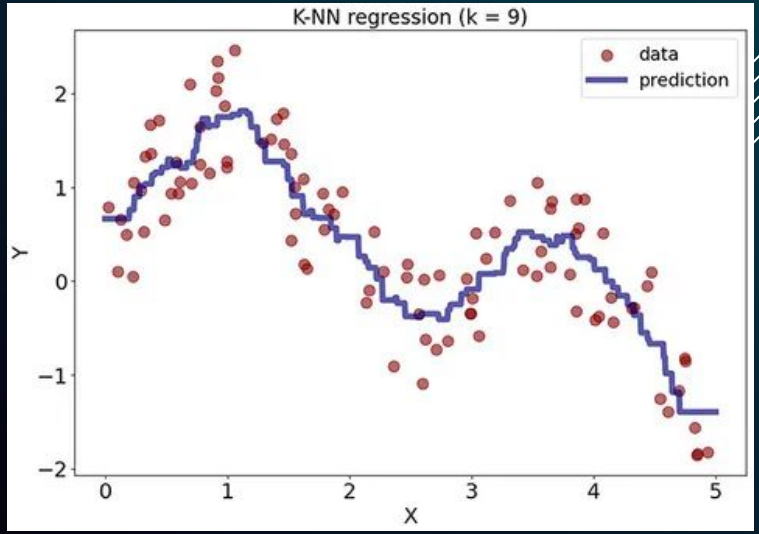


Random Forest

- Many Decision Trees
- Trees “vote”, averaging predictions
- Reduces overfitting and improves accuracy

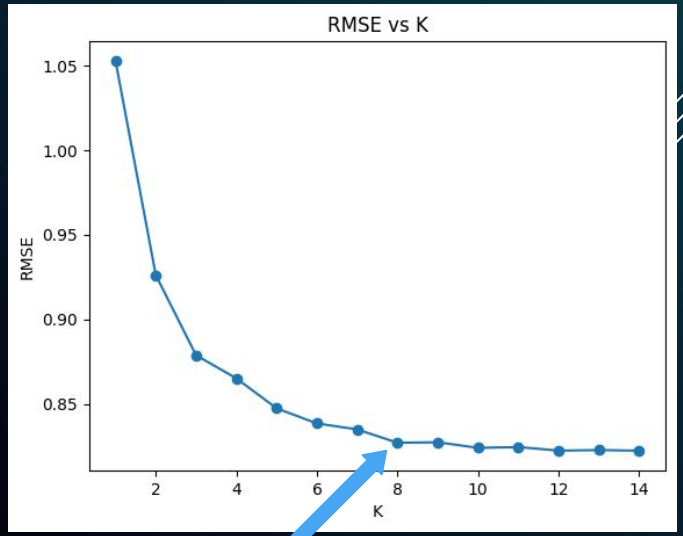
K-Nearest Neighbors (KNN)

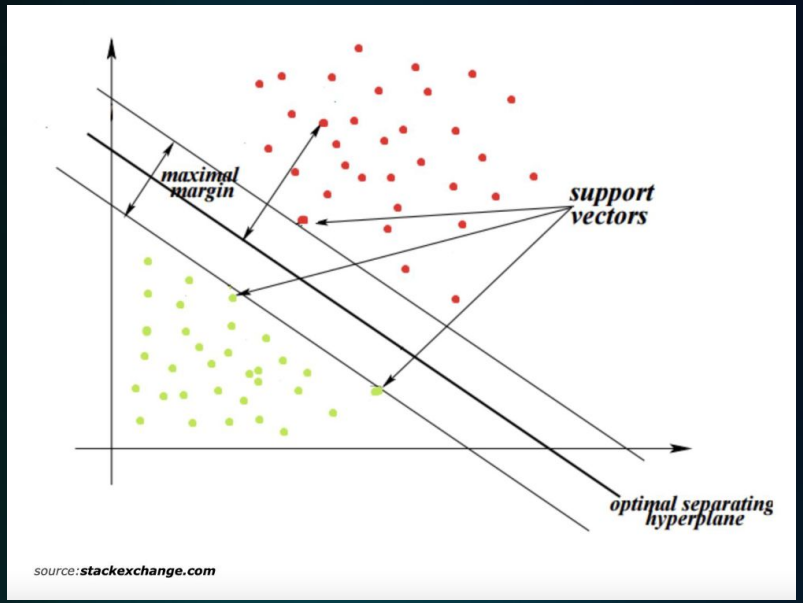
- Predicts based on k nearest data points
- No training phase



K-Nearest Neighbors (KNN)

- Predicts based on k nearest data points
- No training phase
- Chose k = 8 based on the “elbow” point in the RMSE vs K plot



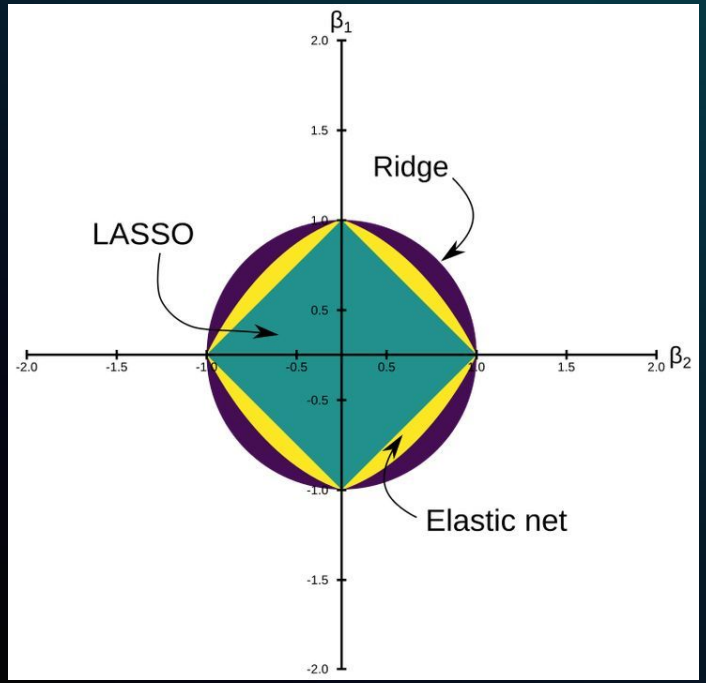


Support Vector Machines (SVM)

- Finds best hyperplane in feature space
- Used Support Vector Regression (SVR)
- Works well with non-linear data

Elastic Net

- Combination of Lasso and Ridge regression
- Selects important features



04

Evaluation and Conclusion

Evaluation

- Used Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R^2
- Used Ten Fold Cross-Validation, averaging the RMSE, MAE, and R^2
- Random Forest has lowest RMSE, MAE, and highest R^2, making it the winner

Best Model

Model	RMSE	MAE	R^2
Random Forest	0.80687	0.56328	0.49483
KNN	0.82704	0.59395	0.46787
SVR	0.85856	0.58717	0.42879
Decision Trees	1.05981	0.70775	0.12605
Elastic Net	1.07299	0.86986	0.10942

Example (Random Forest)

Name	Negative Reviews	Positive Reviews	Number of Reviews	Average Expected Grade	Average Courses GPA	Number of courses	Average Rating	Rating Prediction
Maksym Morawski	48	36	84	2.791626	2.884761	15	2.8079	3.164587
Fawzi Emad	45	93	138	2.736986	2.657434	11	3.6438	3.778388
Allan Yashinski	13	42	55	3.207317	2.809229	33	4.5854	4.076725
Calin Belta	0	4	0	3.844444	3.126531	1	5.0	4.829163
Clyde Kruskal	30	7	37	2.665217	2.772964	16	2.8478	2.580262

Conclusion

- Random Forest was the clear winner
- The positive and negative reviews using Sentiment Analysis were the most important features
- Overall, the models were not horrible
    - For Random Forest, the average prediction was about 0.78 stars off (using RMSE)
- Models might be able to be improved with more or different features
- These models can be used to predict new professors' performances (could be used by the UMD hiring department)