COMP 4432: Machine Learning

Course Overview

This course explores machine learning techniques and theory. The course covers how to use popular machine learning libraries to develop, train, evaluate, and deploy predictive models on prepared data. Both design principles (machine learning types and tasks) and technical tools/languages will be covered.

Objectives

Students will understand and be able to apply machine learning techniques to train, evaluate, and tune models to increase performance, as well as prepare data for analysis via feature engineering. Students will write Python scripts utilizing the following packages: numpy, pandas, matplotlib, Seaborn, scikit-learn, TensorFlow 2, and others as necessary.

Textbooks and Materials

Required textbooks for COMP 4432 Machine Learning:

Géron, A. (2019). *Hands-on machine learning with scikit-learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems* (2nd ed.). O'Reilly Media.

Assignment datasets:

Assignment datasets are located here: https://github.com/arjayit/cs4432_data. Instructors will provide additional guidance when they assign each deliverable. Please reach out to your instructor if you have any questions or trouble accessing the data sets."

Grading

Assignment/Assessment	Points	Weight on Final Grade
Assignment 1	100	20%
Assignment 2	100	20%

Assignment 3	100	20%
Assignment 4	100	20%
Assignment 5	100	20%

Grading Scale

- A = 93–100
- A－ ＝ 90–92.99
- B+ = 86–89.99
- B = 83–85.99
- B－ ＝ 80–82.99
- C+ = 76–79.99
- C = 73–75.99
- C－ ＝ 70–72.99
- D+ = 66–69.99
- D = 63–65.99
- D－ ＝ 60–62.99
- F = < 60

Assignment and Assessment Information

Assignment 1 (due by midnight MST the day prior to Live Session 2)

- **Assignment 1, Part 1**: Data Loading and Preparation. Load the diabetes dataset into two numpy arrays: one for the feature set and one for the target. Pick a single feature to try to predict the target (disease progression). Document the reason you chose the feature you did. Break your single feature and target sets into training and test sets with the last 20 rows being in the test set.
- **Assignment 1, Part 2**: Model Training. Instantiate a linear regression model, and train it with your single feature and target sets.
- **Assignment 1, Part 3**: Prediction and Measurement. List the first 10 predictions on your single feature training set. Print out the feature coefficient and the root mean squared error of your model.
- **Assignment 1, Part 4**: Visualization. Print out a scatter plot with the feature you chose on the x-axis, and progression on the y-axis. Plot the regression line on this same graph with appropriate labels on each axis.

Assignment 2 (due by midnight MST the day prior to Live Session 4)

- **Assignment 2, Part 1**: Data Exploration. Load the scikit-learn diabetes bunch object into a variable. Print out the description of the dataset. Load the diabetes features into a pandas dataframe with the proper column names. Add the target variable to this same dataframe. Run a command to look at the data types of your dataframe to see if there is any missing data. Perform descriptive statistics on the numeric columns of your dataframe. Plot histograms of your data to get a feel for each column's distribution. Split your dataframe into a training and test set with 20% of your data being in the test set. Define a correlation matrix. Look at values highly correlated with the target. Plot the correlation matrix with a Seaborn heatmap. Use a Seaborn pairplot to look at the scatter plots of the three values with the highest target correlation. Prepare a feature set by dropping the target from your training dataframe. Copy your training target into a new dataframe.
- **Assignment 2, Part 2**: Model Training. Train a linear regression model using your training set. Print the RMSE of your regression model on your training set. Implement a cross_val_score on a decision tree regressor on your training set. Print out root mean and standard deviation of the cross-validation scores. Do the same for a RandomForestRegressor. Record which model performs better.
- **Assignment 2, Part 3**: Model Tuning. Print out the parameters of your random forest model. Do a grid search cross-validation with the following values: n_estimators: 3,10,30 and max_features: 2,4,6,8, as well as the following experiment: bootstrap: False, n_estimators: 3,10 and max_features: 2,3,4. Print out the best parameters and the best performing model based on this grid search. Using the cv_results dictionary, print out the rmse of each feature combination for comparison. Also print out the feature importances of the best performing grid search model. Describe how it compares with the correlation matrix we implemented earlier.
- **Assignment 2, Part 4**: Model Evaluation. Document the best-performing model between the single feature model you trained in Assignment 1, and the models you trained in part 2 and 3 of this assignment. Evaluate the best performing model against your test set. Save your model for future use.

Assignment 3 (due by midnight MST the day prior to Live Session 6)

- **Assignment 3, Part 1**: Data Exploration. Load the titanic dataset from Seaborn by using the *load_dataset('titanic')* method. Document the columns that are missing data both numerically (via a count) and visually (via an sns heatmap). Document which values are categorical. Explore the data and answer the following questions: Did more women or men die on the *Titanic*? Which passenger class was more likely to survive? What does the distribution of fare look like? What does the distribution of non-null age values look like? What is the median age of each passenger class (pclass)? Visualize this in a box plot.

- **Assignment 3, Part 2**: Data Cleansing. Since there are so many missing values in Cabin, get rid of the cabin feature. Define a function to impute age using the median of the passenger class you computed earlier. To call it, use *train[['age', 'pclass]].apply(impute_age,axis=1)*. Drop the remaining records containing null values. Show there are no remaining null values. Convert categorical variables to numeric dummies using pandas' *get_dummies()* method. Add these to your training dataframe. Drop the categorical columns you converted earlier as well as *name, ticket, and passengerId*. Create a feature set by dropping "Survived." Your resulting feature set should include pclass, age, sibsp, parch, fare, and the categorical dummy columns you created earlier. Implement a label dataframe by copying the contents of the Survived column of your training set to a new dataframe. Split your clean data into a training and test set.
- **Assignment 3, Part 3**: Model Training. Implement a logistic regression model. Implement a support vector classifier. Implement an sgd classifier. Print out the classification reports, confusion matrices, and roc score and chart for each of these. Remember to set Probability=True for SVM and use method=decision_function in a cross_val_predict instead of predict_proba for the SGD ROC plot.
- **Assignment 3, Part 4**: Model Tuning
  - See if scaling your input data affects your SVC model (implement a sklearn pipeline to combine scaling and instantiation of your model).
  - Do a grid search of your pipeline classifier using the following parameter grid: {'<your_svc_model_name>__kernel': ['rbf'], '<your_svc_model_name> __gamma': [0.0001, 0.001, 0.01, 0.1, 1], '{'<your_svc_model_name>__C': [1,10,50,100,200,300]}.
  - Print the best estimator, its parameters, and the resulting score. Apply this estimator to your test set
  - Implement a learning curve using your best estimator from the grid search.
    - The figure should have a title of "learning curve."
    - Label the y-axis with "Score."
    - Label the x-axis with "Training Examples."
    - Make the training score red.
    - Make the validation score green.
    - What does this learning curve tell you?

Assignment 4 (due by midnight MST the day prior to Live Session 8)

- **Assignment 4, Part 1**: Data Exploration
  - Read in bike_share_hour.csv as a pandas dataframe. The columns are described in the bike_share_readme.txt if you need more information about them.

   - Look at the dataset, and convert the columns that are categorical to a pandas “category” type.
   - Look for non-null values in the dataset.
   - Do a descriptive analysis of the numeric columns.
   - Implement a bar plot of cnt versus season. Document which season has the most bike rides and which season has the least.
   - Implement a bar chart for working day versus count. Document how bike rides are distributed across these two classes.
   - Implement a bar chart for month versus count. Document which months have the most bike rides.
   - Implement code to figure out which months belong to which seasons.
   - Implement a bar plot of weathersit versus cnt. Document which weather situation has less bike rentals.
   - Implement a point plot of weathersit on the x-axis, count on the y-axis, and the season as the hue. Document how season and weathersit are related.
   - Implement a bar plot of hour versus count. Are there any specific hours that are busier than others?
   - Implement a bar plot of hour versus count on weekends and holidays (when workingday = 0). Does the hourly trend change on weekends?
- **Assignment 4, Part 2**: Data Preparation
   - Implement and graph a correlation matrix with the remaining numeric features. Any interesting relationships?
   - Scale the numerical features using StandardScaler(), and replace the original columns in your dataframe.
   - Drop the following columns from your dataset: casual, registered, dteday, instant.
   - Implement a histogram of the count column. What can be said based on the resulting distribution?
   - Implement a train/test split with a test size of 33%.
   - Implement a baseline linear regression algorithm. Use cross-validation to output r2 and mse. Calculate RMSE base on mse. Document your scores.
- **Assignment 4, Part 3**: Model Training (Hint: trained all of these with a for loop and added my results to a PrettyTable.)
   - Create one-hot-encoded values for your categorical columns using get_dummies and add them to your source dataset.
   - Drop the original categorical columns from your source dataset.
   - Do a test/train split based on your new source dataset. Implement and fit a new linear model on your new training set.
   - What are the new values for r2, mse, and rmse?
   - Implement and score a decision tree regressor with random_state=0.
   - Implement and score a RandomForestRegressor with random_state=0 and n_esitmators=30.
   - Implement and score an SGDRegressor with max_iter=1000 and tol=1e-3).

    - Implement and score a Lasso Regressor with alpha=0.1.
    - Implement and score an ElasticNet Regressor with random_state=0.
    - Implement and score a Ridge Regressor with alpha=0.5.
    - Implement and score a BaggingRegressor.
- **Assignment 4, Part 4**: Model Tuning
    - Take the top three performing models and implement cross-validation on them.
        - Hint: They should be Decision Tree Regressor, RandomForestRegressor, and BaggingRegressor.
    - Take your top performing model (mine was the RandomForestRegressor) and do a randomize search cv with 20 iterations and three folds.
        - I found it is best to set your n_jobs = (# of cpu's you have – 1). This took about 10 minutes on my MacBook with 4 CPUs and 8 GB of memory.
        - Your param distributions should include the following:
            - Bootstrap: true, false
            - Max_depth: 10-110, number of bins 11
            - Max_features: auto, sqrt
            - Min_samples_split: 2,5,10
            - Min_samples_leaf: 1,2,4
            - N_estimators: 200 – 2000, number of bins 10
    - Take your best_estimator_ and see how it compares by doing cross_vals for r2, mse, and calculating rmse.

Finally, run predictions on your test set with this model, and see how your r2 score and RMSE look.

Assignment 5 (due by midnight the day prior to Live Session 10)

- **Assignment 5, Part 1**: Implement a Perceptron
    - Given the diabetes dataset you used during Assignment 2, implement an MLP Regressor.
- **Assignment 5, Part 2**: Implement a Keras Classifier
    - Given the prepared *Titanic* dataset from Assignment 3, implement a Keras sequential classifier with relu activation functions.
- **Assignment 5, Part 3**: Implement a Keras Regressor
    - Given the prepared bike-share dataset from Assignment 4, implement a Keras sequential regressor with relu activation functions.
- **Assignment 5, Part 4**: Tune Your Keras Regressor
    - Tune your Keras regressor from Part 3 by implementing a grid search with different optimizers.

Weekly Schedule

There will be a graded assignment assigned each odd week and due the following week by midnight the day prior to the live session. The schedule also includes many asynchronous exercises in addition to the assignments. Please complete each week's asynchronous exercises 24 hours before each live session.

Week 1. Machine Learning Basics

Readings:

- Reading 1: Géron, Chapter 1

Week 2. Data Analytics Project and Process Management

Readings:

- Reading 1: Géron, Chapter 2

Complete Assignment 1

Week 3. Classification

Readings:

- Reading 1: Géron, Chapter 3

Week 4. Model Training

Readings:

- Reading 1: Géron, Chapter 4

Complete Assignment 2

Week 5. Support Vector Machines

Readings:

- Reading 1: Géron, Chapter 5

Week 6. Decision Trees

Readings:

- Reading 1: Géron, Chapter 6

Complete Assignment 3

Week 7. Ensemble Trees

Readings:

- Reading 1: Géron, Chapter 7

Week 8. Artificial Neural Nets

Readings:

- Reading 1: Géron, Chapter 10

Complete Assignment 4

Week 9. Training Deep Neural Networks
Readings:
- Reading 1: Géron, Chapter 11

Week 10. Custom Models in TensorFlow
Readings:
- Reading 1: Géron, Chapter 12

Complete Assignment 5

Attendance Policy
Attendance at all live session meetings is mandatory.

Program Mission
Our MS in Data Science provides students with a broad course of study in programming, algorithms, statistics, and data management, as well as a depth of understanding in specific fields such as data mining, machine learning, and parallel systems. Graduates of the data science program go on to work in a wide variety of careers, including business, government, education, and the natural sciences.

Honor Code and Academic Integrity
All students are expected to abide by the University of Denver Honor Code. These expectations include the application of academic integrity and honesty in your class participation and assignments. Violations of these policies include but are not limited to

- Plagiarism, including any representation of another's work or ideas as one's own in academic and educational submissions
- Cheating, including any actual or attempted use of resources not authorized by the instructor(s) for academic submissions
- Fabrication, including any falsification or creation of data, research, or resources to support academic submissions

Violations of the Honor Code may have serious consequences including, but not limited to, a zero for an assignment or exam, a failing grade in the course, and reporting of violations to the Office of Student Conduct.

Diversity, Inclusiveness, Respect

DU has a core commitment to fostering a diverse learning community that is inclusive and respectful. Our diversity is reflected by differences in race, culture, age, religion, sexual orientation, socioeconomic background, and myriad other social identities and life experiences. The goal of inclusiveness, in a diverse community, encourages and appreciates expressions of different ideas, opinions, and beliefs, so that conversations and interactions that could potentially be divisive turn instead into opportunities for intellectual and personal enrichment.

A dedication to inclusiveness requires respecting what others say, their right to say it, and the thoughtful consideration of others' communication. Both speaking up AND listening are valuable tools for furthering thoughtful, enlightening dialogue. Respecting one another's individual differences is critical in transforming a collection of diverse individuals into an inclusive, collaborative, and excellent learning community. Our core commitment shapes our core expectation for behavior inside and outside of the classroom.