COMP 4220 Machine Learning Final Project, Fall 2021
Due date: December 17, 11pm

This project is an opportunity for you to explore ideas that you see in the lectures, assignments, and other resources. You can think of your project as the first step towards doing research in machine learning. You will analyze the problem, design a machine learning solution, implement learning algorithms, and evaluate them on two data sets (**one for classification and one for regression**). Please review the full list of evaluation metrics at https://scikit-learn.org/stable/modules/model_evaluation.html. In this project, you are asked to (1) develop models that attain high accuracy and (2) explain the performance of the trained models. For example, let's say you designed a classifier with 99% accuracy. An important task is to investigate whether the studied data set is too simple, or the evaluation metric is not appropriately capturing the classifier performance (e.g., we discussed precision, recall, and ROC curves for classification problems). Recall that we should always use training, validation, and test data sets to properly evaluate the performance of machine learning algorithms to avoid underfitting and overfitting.

For each data set, you should use **three** machine learning algorithms. For example, for solving a regression problem, you may use linear regression, polynomial regression, regularized regression (either linear or polynomial). In this case, you should discuss how you tuned the regularization parameter with some evidence. For developing classifiers, we covered a variety of classification algorithms, such as logistic regression, support vector machines, decision trees, random forest, and neural network models. Also, you should think of utilizing preprocessing techniques, such as centering and scaling, to improve the performance of learning algorithms.

Milestones and Grading

The project is worth 20% of the class grade. The final report should be **five pages**. The report should be structured like a small research paper. Broadly speaking it should describe:

- What are the important ideas/methods you explored?
- Preprocessing techniques?
- Reporting the results (cross-validation and easy-to-read figures).
- Do the results make sense? Underfitting? Overfitting?
- Explain the behavior of models (e.g., does the model outperform a random classifier?)
- Please include the complete execution code to produce the reported results at the end of your report. (No page limit)

You will be assessed on the effort, the clarity of explanations, the evidence that you present to support your claims, and the performance of machine learning methods.

Data Sets

- Classification
  - Credit Card Fraud Detection
  - https://www.kaggle.com/mlg-ulb/creditcardfraud?select=creditcard.csv
  - Required to use the imbalanced-learn package: https://imbalanced-learn.org/stable/index.html (tutorial available at https://youtu.be/583SOiipnQ4)
- Regression
  - Energy Efficiency Data Set
  - https://code.datasciencedojo.com/datasciencedojo/datasets/tree/master/Energy%20Efficiency

An example of the final report from previous offerings of this course has been provided. Please note that requirements change every semester, and the attached document solely serves as an example.

I. Introduction

Throughout this course, we were introduced to many machine learning techniques for use in both classification and regression problems. In this paper, we set out to apply these methods in the analysis of sample data sets provided by Scikit-learn. In particular, we present results of 86% accuracy using Support Vector Machines on the Labeled Faces in the Wild data set. Furthermore, we compare the accuracy of a variety of prediction methods on the Breast Cancer data set, namely the Decision Tree Classifier, Logistic Regression, Support Vector Machines, and a Multi-Layer Perceptron Classifier (Neural Network). In this case, we report accuracies of 93%, 98%, 99%, and 99% respectively. Finally, we compare the performance of linear regression to that of Neural Networks on the Boston Housing Data set. In this case, we report mean squared error values of 21.8 and 20.15 respectively.

II. Large Data set: Labeled Faces in the Wild

A. The data set

For our large data set, we chose to use the Labeled Faces in The Wild. We formulated our problem as a facial recognition problem. That is, given an existing database of images (our training set), we wish to report the name of the person in a given input image. Note that it is important to stress the difference between the problem of face recognition and the problem of face detection. The former assumes that the input image necessarily contains a face and tries to detect, from a group of known faces, which face it is, while the latter does not make this assumption, and instead tries to detect weather there exists a face in the input image at all. For the purpose of this project, we concern ourselves only with the problem of face recognition.

B. Our Initial approach

We started by first finding some vector representation of the images in the data set. To do so, we preformed a linear transformation, $f : \mathbb{R}^{m\times n} \to \mathbb{R}^{mn\times 1}$ where f essentially "flattens" out an image into a single row vector of pixels by concatenating the rows together. Now that we have a vector representation of our images, we needed a way to express similarity between two images. For this, we used the cosine distance between two vectors. That is, let u and v be the vector representations of two images in our data set. The measure of similarity between them can be expressed as:

$$\text{Similarity} = cos(\theta) = \frac{\vec{u}\cdot\vec{v}}{||\vec{v}||||\vec{u}||}$$

which follows from the definition of the inner product. Therefore, our initial strategy was a follows: given an input image, transform it into a vector, then compute its cosine similarity pair-wise between every other image in the database. This strategy was okay for small data sets, but presented big problems as the training set grew. The first problem is the obvious one- performance at scale. The performance of this approach depends on both the size of the data set as well as the size of the images within it. We tried to deal with the latter by simply re-sizing the images, however as the LFW data set contains 13,000 labeled images from 5749 people, it was not possible to decrease the size of the data set without also decreasing our accuracy. The second problem is more subtle and has to do with the noise in the images. We found that many of the pixels in the image were actually not contributing much information towards recognizing a face. These were usually pixels around the face representing the background but not telling us anything about who was in the image. This really impacted the accuracy of our approach because the measure of cosine similarity was taking into consideration the background of the image where two images were somewhat similar if they had a similar background even though they had different faces! Clearly we needed a new approach.

C. The Revised Approach

After our first approach failed, we did some more research and came across the eigenfaces method. The main idea here is that instead of using all of the pixels in each image, we can use the dimensionality reduction technique, PCA, to map our vectors from $\mathbb{R}^{mn\times 1}$ to $\mathbb{R}^{k\times 1}$ where $k < mn$. In our implementation, we found that $k = 125$ struck the perfect balance between accuracy and performance. One pitfall that we had here was that initially we fit the PCA model on the entire data set. This of course was a mistake since it resulted in over fitting, as our algorithm was essentially memorizing the data. To fix this issue, we went back and applied PCA to the training set only, leaving the test set untouched. Dimensionality reduction here solved much of the problems we saw in our initial approach. First, reducing the number of features significantly boosted performance since we no longer have to waste time doing computations that don't give us any information. However, more importantly, in reducing the number of features, we also reduced the complexity of our model, as we learned when we covered the bias-variance trade-off in class.

D. Training and Validation

After running PCA, it was time to train our model. We chose to use Support Vector Machines, as they have been proven to be effective at such tasks. We used Scikit-learn's "GridSearchCV" function to select our hyper parameters, which resulted in a regularization term of $C = 1000$ and $\gamma = 0.001$. In particular, to interpret this value of C, we note that higher values of C imply a more complicated decision boundary, which suggests that our decision boundary was highly non-linear. To validate our model, we used our previously trained PCA model to transform the test set to our current feature space. Then we used the SVM model to classify the faces in the test set. A sample of the results can be seen in Figure 1.

E. Analysis

As table I suggests, the accuracy of our results is about 85%. While this does not seem too great compared to what we achieved with the other data sets, it's important to note both the simplicity of the model as well as the amount of training data. One reason for this result is simply that we may need more training data. Initial tests suggest that adding more examples to the training set would increase the accuracy to about 90%. While this model is quite effective, we did notice that it was fairly sensitive to changes in intensity (brightness). Since the images were black and white, there were some input images which were so overshadowed that we could hardly even make out the faces ourselves. Secondly, the model is also sensitive to the angle of the face in the image. That is, we noticed many of the errors were because the input image was pointed one way but the actual images in the training set were pointed in another. Therefore, we think that controlling for the brightness of the images as well as the angle of the face in the image would also help increase the success of our model.

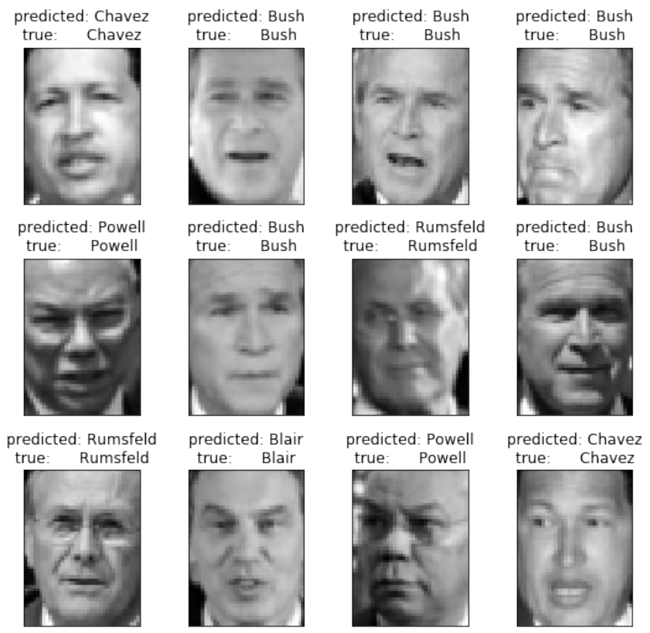

Fig. 1. Face Detection Results

TABLE I
FACES DATASET CLASSIFICATION REPORT

	precision	recall	f1-score	support
Ariel Sharon	0.90	0.75	0.82	24
Colin Powell	0.80	0.91	0.85	58
Donald Rumsfeld	0.87	0.76	0.81	34
George W Bush	0.83	0.93	0.88	129
Gerhard Schroeder	0.86	0.75	0.80	24
Hugo Chavez	0.92	0.67	0.77	18
Tony Blair	0.93	0.74	0.83	35
micro avg	0.85	0.85	0.85	322
macro avg	0.87	0.79	0.82	322
weighted avg	0.85	0.85	0.85	322

III. CLASSIFICATION DATASET: BREAST CANCER DATASET

A. The Dataset

The next data set we selected was the breast cancer data set to explore possible models to classify if a given tumor is benign or malignant. This data set, originally from the University of Wisconsin, consists of 30 features and 569 instances. The attributes set contains the radius, texture, perimeter, area, and more about a given tumor.

B. *Data Exploration*

The first step we took in working on a new data set is to explore it. We used a series of plots to visualize and understand the data better.

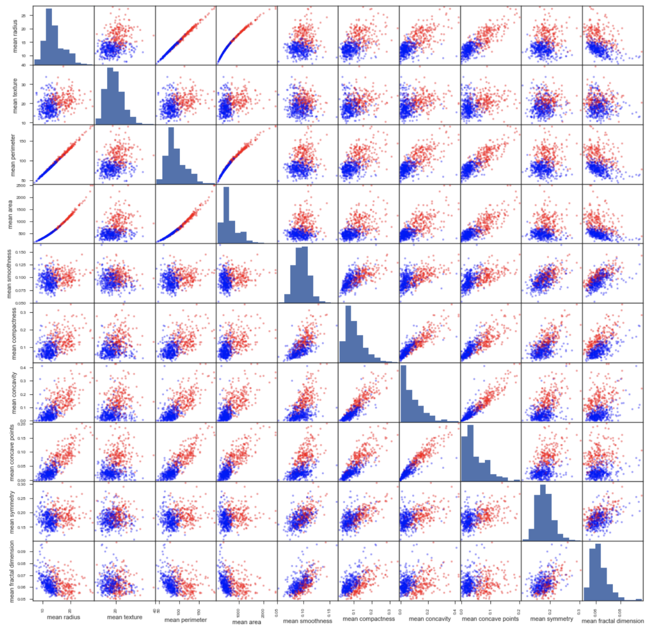

Fig. 2. Features Pair Plot

We used a heat map to observe the correlation between features and discovered both low and high correlation between features. In particular, we found mean compactness and mean concave points to be of interest. In addition to the heat map, we used a pair plot to visualize a feature-by-feature plot, two features at the time. Note that we assign different colors to differentiate benign vs malignant.

C. *Pre-Processing the data*

After visualizing the data, we use Scikit-learn to normalize the data set. The main reason for this is that the features were on completely different scales, so for example, in computing the Euclidean distance between data points those features with a high magnitude will weight higher relative to the other features with low magnitude. The Standard Scaler from Scikit-Learn, uses a straightforward formula, $z = \frac{x-\mu}{s}$, where x is the value to be scaled, μ is the mean of the samples, and s is the standard deviation. This is great because as shown in class, models can produce much better results after normalization.

D. *Comparing Different Models*

Although not mentioned in class, we started exploring by using a simple method, a Decision Tree Classifier. In a Decision Tree, leaves represent a class label and branches represent conjunctions of features that lead to those labels. We were surprised by the outcome of this model. Although not perfect, this model scored a precision, recall, and f1-score of 93% percent.

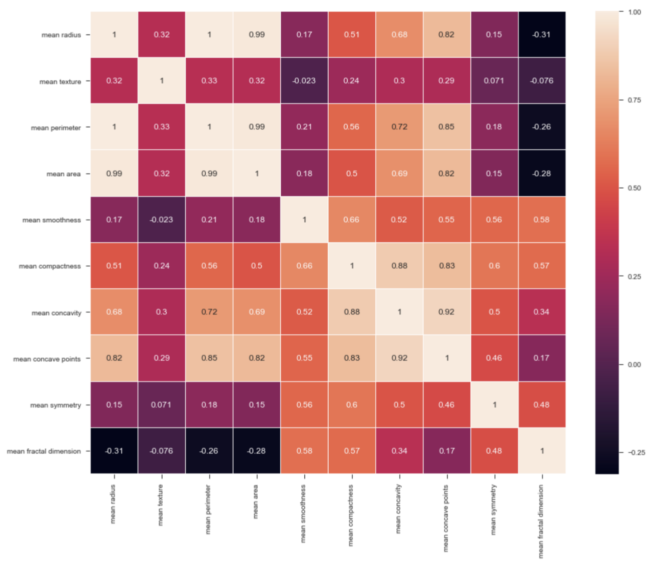

Fig. 3. Correlation Heat map

TABLE II
DECISION TREE CLASSIFICATION REPORT

	precision	recall	f1-score	support
WDBC-Malignant	0.89	0.92	0.90	61
WDBC-Benign	0.95	0.94	0.95	110
avg / total	0.93	0.93	0.93	171

Another method that we applied was a Logistic Regression model, which showed much more improvement. This model scored a precision, recall, and f1-score of 98% which is beat our expectations, given the simplicity of Logistic Regression. Yet, because we are talking about cancer, we were eager to explore more sophisticated methods that we learned in class.

TABLE III
LOGISTIC REGRESSION CLASSIFICATION REPORT

	precision	recall	f1-score	support
WDBC-Malignant	0.98	0.97	0.98	61
WDBC-Benign	0.98	0.99	0.99	110
avg / total	0.98	0.98	0.98	171

Searching for better answers, we used a Support Vector Machine model in conjunction with a Grid Search that helped us tune the hyper-parameters of the SVM. We provided a set of parameters to the Grid Search object to determine the best estimator. Our results concluded that best estimator is one with a $C = 10000$, a $\gamma = 0.0001$, and a radial basis kernel function. Surprisingly, our report showed a result worse than the one shown by the Logistic Regression model scoring a 95%. Because of that, we decided to further tune the value for γ and C. We found a much better result with a $\gamma = 0.00001$, scoring 98% for all, precision, recall, and f1-score.

TABLE IV
SVM Classification Report

	precision	recall	f1-score	support
WDBC-Malignant	0.98	0.97	0.98	61
WDBC-Benign	0.98	0.99	0.99	110
avg / total	0.98	0.98	0.98	171

To take our comparison a step further, we decided to up the complexity and use a Multi-Layer Perceptron Classifier (Neural Network). We played around with different architectures and came out with better results than all of the previous models. Using a hidden layer setup of $(20, 10, 30)$, our neural network was able to score 99% for precision, recall, and f1-score.

TABLE V
MLP Classification Report

	precision	recall	f1-score	support
WDBC-Malignant	1.00	0.97	0.98	61
WDBC-Benign	0.98	1.00	0.99	110
avg / total	0.98	0.99	0.99	171

E. Analysis

Overall, the results for all of the models were pretty accurate. As we expected, the models became more accurate as they grew in sophistication. This can be seen with the Neural Network being the highest out of all them. What we found interesting, however, was that Logistic Regression, even with all of its simplicity was able to compete with the likes of SVM and the Neural Network. Prior to carrying out this comparison, we expected it to be least successful, however this speaks to the power of logistic regression. In other to improve these models one can keep exploring the data and try to find that don't have a high correlation. Given more time, one can find a method to reduce the dimensionality of the data, i.e., removing some features that don't have much impact on the classification problem.

IV. Regression Data Set: Boston Housing Data set

A. The Data set

For our next data set, we chose to use the Boston Housing Data Set for regression. The data set contains roughly 506 samples with 13 features in total. Our objective is to predict the prices of a given house (continuous value) using the features in data set.

B. Exploring The Data

Given the number of features in this data set, it was hard to know which features were well-correlated and which were not. To find out, we used the heat map function in seaborn to visualize the correlation matrix between all of the features in the data set. Additionally, we created a pair plot to help us understand the best set of features to that exhibited a linear trend. Through this exploration, we found that the number of rooms (RM) and the percentage lower status of the population (LSTAT) are the two features that were the most correlated with the price of housing.

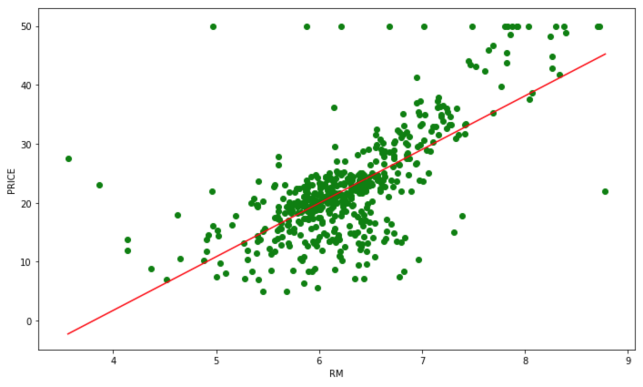

Fig. 4. Price vs Room Numbers

C. Data Pre-Processing

Next, we took the Boston housing data set and split the data into training and testing subsets. Typically, the data is also shuffled into a random order when creating the training and testing subsets to remove any bias in the ordering of the data set. Next, was the normalization step. The Boston Housing data set contains many features whose scale differs from one another. For example, the feature CRIM ranges between 0 and 1 while Age ranges from 0 to 100. We discovered that when we did not normalize the data we ended up with a very high Mean Squared Error (MSE). We normalized the data by subtracting the mean of the feature and dividing by its standard deviation to make the data between 0 and 1.

D. Linear Regression

The first model, we tried was the simple Linear Regression Model. This was the first technique we learned in class, and as we saw in the breast cancer data set, simple models can sometimes compete with more sophisticated ones. To judge the performance of our model, we used the regression evaluation matrix to calculate the mean squared error. One mistake we made, was that we tried to fit the linear regression model before we normalized our data. This gave us an MSE of 28.54, in which case we thought for sure that this model was ineffective. However, after normalizing the data set, our mean squared error went down to a more acceptable 21.89.

E. Deep Learning: Neural Network

Next, we used the Keras API along with TensorFlow to build and train a Neural Network for predicting housing price. After importing the data set, we shuffled it so that later when we split the data set between training and testing sets, the data will be distributed nicely. Then we split the data set into testing and training set such that our training set contained 404 rows and the testing set contained 102 rows.

We used the Sequential model since it was highly configurable and allowed us to build the model layer by layer. We used the model.add() function to add 3 densely connected hidden layers in which the first layer consisted of 50 nodes, the second layer consisted of 100 nodes and the third layer consisted of 50 nodes. Finally, we had an output layer consisting of one node which represented the model's output prediction. We used the Rectified Linear Unit (RELU) for the activation function along with the ADAM stochastic optimizer. Furthermore, we used the Mean Squared Loss loss function and ran it for 50 epochs. We noticed that when we used a higher value for epoch, it took longer but did very little to increase the model's accuracy. We then used the model.fit and model.evaluate functions to see how the model performed on the test data set. Our model resulted in a mean squared error of 19.3. To put this value into perspective, when we used only two hidden layers and far fewer nodes, the MSE was around 50 and when we did not normalize the data, the MSE was around 75. Next, we used model.predict() to predict the housing prices using data in the testing set. We calculated an r^2 value of 0.76 which shows that 76 percent of the variance in the real price is explained from by predicted price. Figure 4 shows a comparison of the real and predicted housing prices.

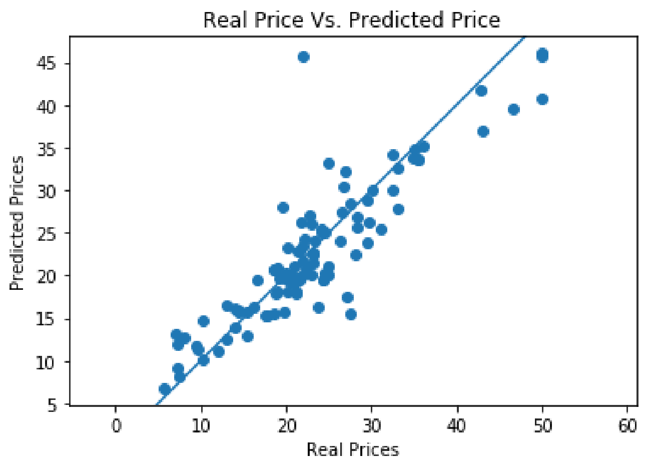

Fig. 5. Real Price vs Predicted Price

F. Comparison

In comparison, the Linear Regression method produced a Mean Squared Error of 21.89 while the Neural Network yielded a Mean Squared Error of 19.3. As we saw with the breast cancer data set, our accuracy grew as the sophistication of the model we used grew. However, the simple model this time Linear Regression, was not too far behind. We suspect this is due to the fact that the data in this case was very well correlated, therefore Linear Regression should have no trouble fitting a simple line through it. We anticipate that the power of Neural Network will supersede that of Linear Regression when dealing with more complex, highly-nonlinear relationships within data.

V. Future Works

Given that our time here was limited, we propose a few extensions to our work in the future. For the faces data set, we would focus on increasing the complexity of our model in order to see precision levels higher than 86%. Our research suggests that Convolutional Neural Networks have proven to be effective at facial recognition, so we'd be interested in exploring that avenue. In the cases of the breast cancer and housing data sets, we would seek to find out the point at which more sophisticated methods like Neural Networks surpass simpler methods like linear and logistic regression. We would experiment with more complex highly non-linear data where the true power of these methods is needed to achieve any reasonable results.