Homework 1: ML 80629A (Winter 2023)

Instructions:

- Please include your name and HEC ID with submission.
- The homework is due by 11:59pm on the due date.
- The homework is worth 10% of the course's final grade.
- Assignments are to be done individually.
- All code used to arrive at answers is submitted along with answers. You can convert your jupyter notebook or colab to pdf and upload it.

1 ML Fundamentals (10pt)

1. (2pt) Does a good training error mean that the model will perform well on new data? Explain your answer.
2. (2pt) What happens if you use the same dataset for both validation and testing your model?
3. (2pt) You chose a large value of k in your k-nearest neighbors algorithm. Is your model overfitting or underfitting? Which case of the bias/variance trade-off is this? Suggest a way to improve the model's performance.
4. (2pt) You have trained a linear regression model for a particular data set, and you observe the coefficient of one of the features having a relatively high value. Can you make any conclusions?
5. (2pt) What is the key assumption behind the Naive Bayes classifier?

2 Regression (30 pts)

- Let's explore Metro Interstate Traffic Volume Data Set. You can load the data from into a pandas DataFrame using the URL directly. The goal is to predict the traffic volume using weather data. In this first part, you will explore the dataset, apply some transformations to it, and train a linear regression model. You will also try different regularizers and see how they can improve the model.

    1. (3pt) Explore the dataset using `.describe()`, and `.info()`. Which features are unsuitable for this regression task? Remove the unsuitable columns from the dataframe.
    2. (3pt) Look at the distribution of the attribute values by plotting their histograms. What do you notice?
    3. (2pt) Split the dataset intro training and test sets using sklearn's *train_test_split* function, with *test_size*=`0.3`, *random_state*=`42`.

- **Target transformation:** In this part of the homework, you will explore how transforming the target variable can help the training. To this end, you can use `TransformedTargetRegressor`. An interesting example of the wanted effect can be found here.

    4. (3pt) Apply a **TransformedTargetRegressor** using a QuantileTransformer with $n\_quantiles = 90$ with a linear regression regressor on the training set and predict the test set. Explain how this transformation impacts the model.
    5. (3pt) Train a simple linear regression model on the training set and predict the test set. Compare the models with and without transformations using Median Absolute Error (MAE) and MSE metrics.

- Three very popular options in Linear Regression are the Lasso method , the Ridge Regression, and Elastic Net. These models are implemented in sklearn: Lasso, Ridge, and ElasticNet.

  Hint: If you'd like to understand Ridge Regression better, you might want to look at this example.

    6. (3pt) What is the idea behind regularization? How do we pick our choice of hyper-parameter α (in the course slides it is referred to as λ)?
    7. (4pt) Using sklearn KFold function Perform k-fold cross-validation with normal LinearRegression, for $k \in [2, 5, 7, 10]$, (with `shuffle=True` and `random_state=2023`).
    8. (3pt) Apply the same k-fold cross validation on Ridge, Lasso, and report the average MSE for each method on the test set across folds.
    9. (3pt) How do these variants perform compared to linear regression?

10. (3pt) Which regularization methods would be suitable for feature selection? Justify your answer. Hint : look at the attribute weights ($coef\_$) for each of Ridge, Lasso, and ElasticNet.

Classification (50 pt)

In this exercise, you will implement K-Nearest neighbours and a support vector machine to carry out a classification task on the IRIS. You can use sklearn to perform your tests: SVC and KFold classes will be particularly handy.

You can also download the dataset using scikitlearn with the commands:

```
iris = sklearn.datasets.load_iris()
x, y = iris.data, iris.target
```

1. (2pt) After all the required imports in your code be sure to set the `random_state=12345`. Split the dataset into a training set and a test set (use 80% vs 20% for the train-test split). You can use the function `train_test_split` and the KFold method `.split` from sklearn to help you.

3 K-Nearest Neighbours (10 pts)

1. (5pt) Run K-NN on the Iris dataset with K=3 with each of the metrics: manhattan , euclidean, and cosine. Which of these metrics worked best? Hint: You can view the confusion matrix and the accuracy of the model.
2. (5 pt) Using the best metric that you have found, find the best number of neighbours (between 1 and 10) using k-fold cross validation with (k=5).

4 SVM (30pt)

In this section, you will learn about kernel functions for SVM. A kernel function is a function that is applied on the input to transform it before it is fed to the SVM. Data that is not linearly separable can be transformed to a new feature space where every data point is now linearly separable (see Figure 1). But how do we pick this feature map? Kernel functions simplify this process because we do not need to explicitly model the feature transformation to a new space. We can instead use it to implicitly transform the features to a new space and hopefully make the data more linearly separable.

As an example, the kernel function $k(x, y) = (x \cdot y)^2$ implicitly defines a feature map ϕ if we define *phi* by the equation $k(x, y) = \phi(x) \cdot \phi(y)$. The function ϕ transforms the features to a higher-dimensional space (which could be much larger). The kernel function instead only performs operations over

the much lower-dimensional orgininal space. Learning SVMs with them will be much more efficient. For more information about SMVs and kernel functions, you can read this article.

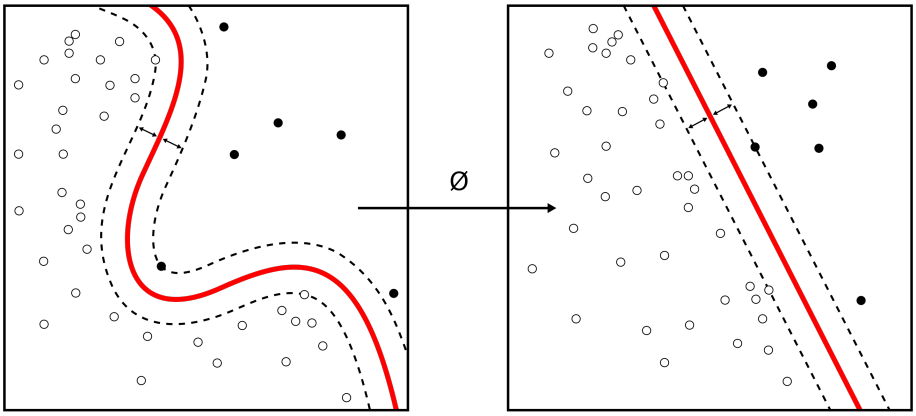

Figure 1: Mapping features to a new space where points are linearly separable

Now, you need to train multiple SVM models using sklearn. For every model, perform k-fold cross-validation ($k = 10$) on the training set.

1. (15pt) You are going to train SVM's with different hyperparameter selections. Use all the following hyperparameter combinations:

   - linear kernel, $C = 0.001$
   - linear kernel, $C = 0.1$
   - linear kernel, $C = 1$
   - polynomial kernel of degree 2, $C = 0.001$
   - polynomial kernel of degree 2, $C = 0.1$

   For each hyperparameter combination, report the training and test loss. What is the best combination of hyperparameters and what is the performance on the resulting model test set (in terms of accuracy of the model)?

2. (15pt) What did you observe with the impact of different hyperparameters on the model accuracy? Write a brief recommendation (i.e. a general method / scheme) to follow for someone trying to find the best hyperparameters for their model. Does using a very large degree for the polynomial kernel always help the generalization error?

5 Comparison (10pt)

1. (5pt) Determine the performance of the simplest baselines, i.e., random classification and majority classification , for this task.

n.b. Remember, majority classification assigns every point to the class with the highest number of training examples. When there is no true majority, this would technically be referred to as a plurality instead.

2. (2pt) Report the best test performance accuracies of the k-NN and SVM that you observe.

3. (2pt) Based on these accuracy values, which is the best performing model? Explain briefly why it is the case.