Lab 2 – ML Engineering

Question 1 – Preprocessing

Let us predict diabetes using k-NN. The diabetes.csv dataset can be downloaded from Moodle or Kaggle (https://www.kaggle.com/uciml/pima-indians-diabetes-database).

1. Load the dataset and have a look at it.
2. Split the dataset into the explanatory variables (X) and the target variable (y). What distance might be used for the k-NN?
3. Perform a train-test split using 60% for the training set and 40% for the test set.
4. Because, we are computing distances, it is preferable for each variable to be on the same scale. Why is that? Use `MinMaxScaler()` to make each column use the [0,1] interval. Pay attention to preventing data leakage by carefully using the `fit()` and `transform()` functions.
5. Train a k-NN model with k=7. Compute the score on the training and test sets.

Question 2 – Curse of Dimensionality

Let us use a synthetic dataset to observe what happens to distances between observations as the number of dimensions increases.

1. Generate 100 observations with 10,000 features distributed uniformly.
2. Select a subset of 2 features, compute all the pairwise distances between the 100 observations and plot the distribution of distances.
3. Perform the previous operation with 3, 10, 100, 1,000 and 10,000 features. Based on the visual comparison of distance distributions, what is your conclusion about the usefulness of distances (for example for k-NN)?

Question 3 – Cross-validation and Parameter Tuning

Let us use logistic regression on the breast cancer dataset. Again, use stratified sampling.

1. Use `GridSearchCV` to evaluate the combination of different values for the following parameters : `solver`, `penalty` and `C`.
2. Once `fit()` is used, the `GridSearchCV` object has multiple interesting attributes: `cv_results_`, `best_estimator_`, `best_score_`, `best_params_`, `refit_time_`. Explore the contents of each of these.
3. Save the model in a file and test that you can load the model back from file.

Question 4 – Pipelines

Let us perform *linear* regression on the Boston dataset (`Boston.csv`). The Boston dataset is used to predict house prices and has 13 features. The target variable is MEDV, and the BIAS_COL column should not be used.

1. Load and split the dataset into training and test sets.
2. We will use a Ridge regression (a classic linear regression using the least squares method, but with L2 regularization). Because we have a linear model, it is better to normalize our data with `StandardScaler()`. We will also do some feature engineering by building polynomial features.
   To do all this, build a pipeline composed of `StandardScaler()`, `PolynomialFeatures()` and `Ridge()`.
3. Specify that the `degree` parameter of `PolynomialFeatures` can be 1, 2 or 3. Also specify that the `alpha` parameter of `Ridge` can be 0.001, 0.01, 0.1, 1, 10 or 100.
4. Use `GridSearchCV` to find the best combination of parameters. Evaluate the score of the model on the test set. Note that the score for regression is the coefficient of determination, R^2. Compare the score to a model not using polynomial features.

Question 5 – Feature Selection and Data Leakage

Let us consider regression on a synthetic dataset. We will generate many features sampled independently from a Gaussian distribution, select a subset of them, and then observe what happens.

1. Generate 100 observations with 10,000 features (Gaussian distribution).
   ```
   rnd = np.random.RandomState(seed=0)
   X = rnd.normal(size=(100, 10000))
   y = rnd.normal(size=(100,))
   ```
2. Even if there is no real information that can be learned from the dataset, let us still try to predict something. Since 10,000 features is too many features anyway, so let us perform feature selection. Use `SelectPercentile()` feature selection directly on X and y, choosing the best 5% of features and using `f_regression` as score function.
3. Split the modified dataset into training and test sets. Then use Ridge regression and compute the score (R^2) on both sets. What is your observation?
4. Now split the original data into training and test sets. Then perform feature selection and Ridge regression. Compute the scores.
5. Now let's use cross-validation on the training set with the selected features. We do not use `GridSearchCV` given that we are not trying to find specific parameter values. We use the `cross_val_score()` function.
   ```
   from sklearn.model_selection import cross_val_score
   cvs = cross_val_score(Ridge(), X_train_selected, y_train)
   np.mean(cvs)
   ```
6. Now use a pipeline on the training data to do the feature selection and fitting the Ridge. Compute `cross_val_score()`. Why is there a difference?