Data Science and Visualization, S2026

Supervised learning 3: Model selection and assessment

2026-04-07

Exercise 1: Importing and loading

In this exercise set, you will use the scikit-learn library to go through the process of selecting, fitting, and assessing a supervised learning model for one regression and one classification problem. You will use two different datasets for this purpose.

The **first dataset** contains information on students enrolled in a **Portuguese upper secondary education** programme. This dataset contains the following columns:

- **school**: Name of the school the student attends (gabriel_pereira, Mousinho_da_silveira)
- **sex**: The gender of the student (female, male)
- **age**: The age of the student (years)
- **address**: Location of the student's home address (urban, rural)
- **famsize**: Number of members of the student's family (3_or_less, greater_than_3)
- **Pstatus**: Cohabitation status of the student's parents (cohabitating, not_cohabitating)
- **Medu**: Educationl level of the student's mother (5-point scale)
- **Fedu**: Educational level of the student's father (5-point scale)
- **Mjob**: Occupation of the student's mother (teacher, health, services, at_home, other)
- **Fjob**: Occupation of the student's father (teacher, health, services, at_home, other)
- **reason**: The student's primary reason for selecting his/her school (close_to_home, school_reputation, course_preference, other)
- **guardian**: The student's guardian (mother, father, other)
- **traveltime**: Home to school travel time (4-point scale)
- **studytime**: Weekly study time (4-point scale)

- **failures**: Number of past courses the student has failed
- **schoolsup**: Does the student receive extra educational support from his/her school? (yes, no)
- **famsup**: Does the student receive educational support his/her family? (yes, no)
- **paid**: Does the student attend extra paid classes in math (yes, no)
- **activities**: Is the student engaged in extra-curricular activities (yes, no)
- **nursery**: Did the student attend nursery school (yes, no)
- **higher**: Is the student aiming to enroll in higher education (yes, no)
- **internet**: Does the student have internet access at home (yes, no)
- **romantic**: Is the student in a romantic relationship (yes, no)
- **famrel**: Quality of the student's family relationships (5-point scale)
- **freetime**: Amount of the free time the student has after school (5-point scale)
- **goout**: Frequency of going out with friends (5-point scale)
- **Dalc**: Volume of alcohol consumption on workdays (5-point scale)
- **Walc**: Volume of alcohol consumption on weekends (5-point scale)
- **health**: Health status of the student (5-point scale)
- **absences**: Number of days the student has been absent from school
- **grade**: The student's grade at the final math exam in upper secondary education

We will use this dataset to develop a model with the purpose of predicting a student's average grade in math throughout upper secondary education. So we will use the column **grade** as the **target**. As grade is quantitative variable, this is a **regression problem**

The **second dataset** contains information on customers of an **Iranian telecom company**. Each row represents a unique customer. All the variables except Churn represent aggregates over a 9 month period. The dataset contains the following columns:

- **Call Failures**: Number of call failures
- **Complains**: Dummy variable equal to 1 if the customer has filed a complaint
- **Subscription Length**: Total months of subscription
- **Charge Amount**: Total amount the customer has been charged (10-point scale)
- **Seconds of Use**: Total seconds of calls
- **Frequency of Use**: Total number of calls
- **Frequency of SMS**: Total number of text messages
- **Distinct Called Numbers**: Total number of distinct phone calls

- **Active**: Dummy variable equal to 1 if the customer is active
- **Age**: Customer age (years)
- **Customer Value**: The calculated value of the customer
- **Churn**: Dummy variable equal to 1 if the customer ended his/her relationship with the company by the end of the 12th month

We will estimate a series a model with the purpose of predicting whether or not customers ended their relationships with the telecom company by the end of the 12th month. So we will use the column **Churn** as the **target**. As **Churn** is a qualitative variable, this is a **classification problem**

The exercise set will ask you to visualize aspects of your models. As always, make sure your plots abide to the principles of data visualization. One approach to controlling the layout of your plots is to specify a global theme with the `theme()` function. You can then modify the theme for each plot you create as needed

Use the empty code cell below to import the NumPy, pandas, and Plotnine libraries and the Lasso, LogisticRegression, KNeighborsRegressor, KNeighborsClassifier, PolynomialFeatures, StandardScaler, roc_curve, roc_auc_score, train_test_split, Pipeline, and GridSearchCV submodules from the scikit-learn library. Download the files *portuguese_students.csv* and *iranian_churn_sample.csv* from Moodle and save them in the same folder as this notebook. Then read the files as DataFrame objects and assign it to them to variable `df_reg` and `df_class`, respectively. Finally, use a DataFrame method to find out how many rows and columns each dataset has and print the results

```
# Import libraries

# Load the datasets from the current directory

# What are the dimensions of the datasets?
```

```
# Specify a global theme to use in all plots
```

Exercise 2: Model selection for a regression problem

2.1 Use the empty code cells below to do the following:

1. Create dummy variables from the all the qualitative features in `df_reg`. Leave out the first category on each qualitative feature, i.e., if there are K classes on a feature, construct dummies for the 2^{nd} through the K^{th} class. You can use the pandas function `pd.get_dummmies()` to do this. Use the list of columns in the Portuguese students data to identify the qualitative features in `df_reg`. Collect the dummy variables in a new DataFrame
2. Create a new DataFrame that contains only the quantitative features in `df_reg`. Use the list of columns in the Portuguese students data to identify the quantitative features in `df_reg`
3. Construct second-degree polynomial terms from the quantitative features in `df_reg` as well as all interactions among them. You can use the scikit-learn function `PolynomialFeatures()` to do this. Collect all the engineered quantitative features in a new DataFrame. Make sure to name all the columns in this DataFrame with strings. The fitted `PolynomialFeatures()` object has a method `get_feature_names_out()` that is useful for this

```
# Create dummies from the qualitative features in df_reg
```

```
# Create a DataFrame with quantitative features in df_reg
```

```
# Create polynomials and interactions from the quantitative features in df_reg
```

2.2 Use the empty code cells below to do the following

1. Assign the target grade from `df_reg` to a new pandas Series `Y_reg`. Then collect the dummy-encoded qualitative features and the original set of quantitative features from `df_reg` without polynomials and interactions in a new DataFrame. Call this DataFrame `X_reg_KNN`. We will use this DataFrame as the feature set for developing a K-nearest neighbors (KNN) regression model for the task of predicting students' math grades. Maybe you are thinking something like: Why do we not include all the 90 engineered quantitative features that we constructed in exercise 2.1 in this feature set?That is a great question for which the answer lies in a concept that we call the *curse of dimensionality*. To can learn more about the curse of dimensionality in James et al. (2023): ISL, pp. 114-115.
2. Use the series `Y_reg`, the DataFrame `X_reg_KNN`, and the scikit-learn function `train_test_split()` to split the available data on Portuguese students into a training set with 80% of the data points and a test set with 20% of the data points

3. Use 5-fold cross-validation and the training set you just constructed to tune a KNN regression model for the task of predicting students' math grades. Choose an appropriate range of values of K and a prediction error metric to base the model selection process on. Make sure to standardize the features at each step of the cross-validation procedure. You will need to use the scikit-learn functions `Pipeline()`, `StandardScaler()`, `KNeighborsRegressor()`, and `GridSearchCV()`
4. When the cross-validation procedure is completed, print the optimal value of K and the cross-validation estimate of prediction error corresponding to the best model. You can find the relevant methods of the `GridSearchCV()` object in the scikit-learn documentation for this function

```
# Assign the target and the features to new variables
```

```
# Split the available data into a training and a test set
```

```
# Use 5-fold cross-validation to tune the KNN regression model
```

```
# Print the results of the cross-validation procedure
```

2.3: Use the empty code cells below to do the following:

1. Collect the dummy-encoded qualitative features and the full set of engineered quantitative features with polynomials and interactions in a new DataFrame. Call this DataFrame `X_reg_LASSO`. We will use this DataFrame as the feature set for developing a LASSO regression model for the task of predicting students' math grades
2. Use the series `Y_reg`, the DataFrame `X_reg_LASSO`, and the scikit-learn function `train_test_split()` to split the available data on Portuguese students into a training set with 80% of the data points and a test set with 20% of the data points
3. Use 5-fold cross-validation and the training set you just constructed to tune a LASSO regression model for the task of predicting students' math grades. Make sure to standardize the features at each step of the cross-validation procedure. Choose an appropriate range of values of λ to base the model selection process on. Use the same prediction error metric that you used for tuning the KNN regression model. You will need to use the scikit-learn function `Lasso()`

4. When the cross-validation procedure is completed, print the optimal value of λ and the cross-validation estimate of prediction error corresponding to the best model. Compare this estimate to the cross-validation estimate of error for the best-performing KNN regression model. The model with the lowest estimated prediction error is your selected model

```
# Assign the features to a new variable
```

```
# Split the available data into a training and a test set
```

```
# Use 5-fold cross-validation to tune the LASSO regression model
```

```
# Print the results of the cross-validation procedure
```

Exercise 3: Model assessment for a regression problem

3.1 Use the empty code cells below to do the following:

1. Fit the model you selected in exercise 2 as the best-performing model for the task of predicting students' math grades *to the full training set*. Make sure to use the right feature set and standardize the features before fitting the model
2. Use the model you just fitted to the training set to predict students' math grades *in the test set*. Make sure to standardize the features before computing predictions. Evaluate the test set performance of the model by computing a range of appropriate prediction error metrics. Assign these estimates of prediction error to a set of new variables

```
# Fit the selected model to the full training data
```

```
# Predict the target in the set and evaluate performance
```

3.2 Report the results of your assessment of the selected model's test set performance by plotting the observed values of the target against the predicted values. Use a scatterplot with a 45-degree through the origin for this purpose. The 45-degree represents a model with perfectly accurate

predictions. Report also your test set estimates of prediction error. To abide with the the third principle of data visualization - *integrate the text and the graph* - display these estimates as an annotation to the graph or in a subtitle

```
# Create a scatterplot of observed and predicted values
```

```
# Output the plot
```

Exercise 4: Model selection for a classification problem

4.1: In the empty code cell below, use the scikit-learn function `PolynomialFeatures()` to construct second-degree polynomial terms of all the quantitative features in `df_class` as well as all interactions among them. Use the list of columns in the Iranian telecom data to identify the quantitative features in `df_class`. Collect this set of engineered features in a new DataFrame. Make sure to name all the columns in this DataFrame with strings. The fitted `PolynomialFeatures()` object has a method `get_feature_names_out()` that is useful for this

```
# Create polynomials and interactions from the quantitative features in df_class
```

4.2 Use the empty code cells below to do the following

1. Assign the target Churn from `df_class` to a new pandas Series object `Y_class`. Then assign the original unaltered feature set from `df_class` to a new DataFrame. Call this DataFrame `X_class_KNN`. We will use this DataFrame as the feature set for developing a K-nearest neighbors (KNN) classifier for the task of predicting customer churn in the Iranian telecom data
2. Use the series `Y_class`, the DataFrame `X_class_KNN`, and the scikit-learn function `train_test_split()` to split the available Iranian telecom customer data into a training set with 80% of the data points and a test set with 20% of the data points
3. Use 5-fold cross-validation and the training set you just constructed to tune a KNN classifier for the task of predicting customer churn. Choose an appropriate range of values of K and a prediction error metric to base the model selection process on. Make sure to standardize the features at each step of the cross-validation procedure. You will need to use the scikit-learn function `KNeighborsClassifier()`

4. When the cross-validation procedure is completed, print the optimal value of K and the cross-validation estimate of prediction error corresponding to the best model

```
# Assign the target and the features to new variables
```

```
# Split the available data into a training and a test set
```

```
# Use 5-fold cross-validation to tune the KNN classifier
```

```
# Print the results of the cross-validation procedure
```

4.3: Use the empty code cells below to do the following:

1. Collect the qualitative features from `df_class` and the full set of engineered quantitative features with polynomials and interactions in a new DataFrame. Call this DataFrame `X_class_LASSO`. We will use this DataFrame as the feature set for developing a LASSO classifier for the task of predicting customer churn in the Iranian telecom data
2. Use the series `Y_class`, the DataFrame `X_class_LASSO`, and the scikit-learn function `train_test_split()` to split the available Iranian telecom customer data into a training set with 80% of the data points and a test set with 20% of the data points
3. Use 5-fold cross-validation and the training set you just constructed to tune a LASSO classifier for the task of predicting customer churn. Make sure to standardize the features at each step of the cross-validation procedure. Choose an appropriate range of values of λ to base the model selection process on. Use the same prediction error metric that you used for tuning the KNN classifier. You will need to use the scikit-learn function `LogisticRegression()`. Note that the argument `C` lets you specify the quantity $1/\lambda$
4. When the cross-validation procedure is completed, print the optimal value of λ and the cross-validation estimate of prediction error corresponding to the best model. Compare this estimate to the cross-validation estimate of error for the best-performing KNN classifier. The model with the lowest estimated prediction error is your selected model

```
# Assign the features to a new variable
```

```
# Split the available data into a training and a test set
```

```
# Use 5-fold cross-validation to tune the LASSO classifier
```

```
# Print the results of the cross-validation procedure
```

Exercise 5: Model assessment for a classification problem

5.1 Use the empty code cells below to do the following:

1. Fit the model you selected in exercise 2 as the best-performing model for the task of predicting customer churn *to the full training set*. Make sure to use the right feature set and standardize the features before fitting the model
2. Use the model you just fitted to the training set to predict customer churn *in the test set*. Make sure to standardize the features before computing predictions. Evaluate the test set performance of the model by computing a range of appropriate prediction error metrics. Assign these estimates of prediction error to a set of new variables

```
# Fit the selected model to the full training data
```

```
# Predict the target in the set and evaluate performance
```

5.2 Use the empty code cells below to do the following:

1. Use a separation plot to report the result of your assessment of the selected model's test set performance. If needed, you can go back to exercise set 7 and brush up on separation plots. Report also your test set estimates of prediction error. To abide with the the third principle of data visualization - *integrate the text and the graph* - display these estimates as an annotation to the graph or in a subtitle. If you have constructed a confusion matrix, output it in a separate
2. To report an assessment of test set performance that is agnostic to decision rules, plot the test set ROC curve. Display the area under the curve on the plot

```
# Create a separation plot
```

```
# Output the separation plot
```

```
# Create a ROC curve
```

```
# Output the ROC curve
#roc_curve
```