Data Science and Visualization, S2026

Supervised learning 2: Classification

2026-03-24

Exercise 1: Importing and loading

In this exercise set, you will use the the scikit-learn library to fit logistic regression models and K-nearest neighbors (KNN) classifiers to customer data from 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
- **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 throughout the exercise set, we will use the column **Churn** as the **target**. The exercise set will also ask you to visualize aspects of the data or 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 LogisticRegression, KNeighborsClassifier, PolynomialFeatures, StandardScaler, roc_curve, and roc_auc_score submodules from the scikit-learn library. Download the file *iranian_churn_sample.csv* from Moodle and save it in the same folder as this notebook. Then read it as DataFrame object and assign it to the variable `df`. Finally, use a DataFrame method to find out how many rows and columns the dataset has and print the results

```
# Import libraries

# Load the iranian churn data

# How many rows and columns does df have?
```

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

Exercise 2: Visualizing distributions and amounts

Before we fit any supervised learning models to the Iranian telecom data, let's use visualization to investigate the relationship between the target Churn and some of the features. The exercises below asks you to do this first for two quantitative features and then for two qualitative features

2.1 Use density plots to visualize the associations between the target Churn and the quantitative features Subscription Length, Seconds of Use, Frequency of Use, and Customer Value. You can do this by following these two steps:

1. Start by constructing a DataFrame containing the data to be plotted. You will need to construct a DataFrame where the values of the quantitative features Subscription Length, Seconds of Use, Frequency of Use, and Customer Value are contained in a single column. A second column should identify which feature the value in a particular row pertains to. A third column should then map the value of the target to the value of the feature displayed in a particular row. You can use pandas method `.melt()` to reshape and filter `df` to have this structure

2. Create a density plots of the features for each value of the target Churn. Map each of the features to small multiples with the function `facet_wrap()`. Use the function `geom_density()` to plot the data. Map the target to the `fill` aesthetic to create separate density plots for churns and non-churns

```
# Step 1: Construct a DataFrame containing the data to be plotted
```

```
# Step 2: Create the plot
```

2.2 Use a stacked bar plot to visualize the associations between the target Churn and the qualitative features Complains and Active. You can do this by following these two steps:

1. Start by constructing a DataFrame containing the data to be plotted. You will need to compute the churn rate across the values of the qualitative features Complains, Contract, and Active. You can use the pandas methods `.group_by()` and `.value_counts()` to do this. Compute the the churn rates separately for each qualitative features and then combine the results into one DataFrame using the function `pd.concat()`. Transform the proportions to percentages and round the values to 1 decimal place
2. Create the plot. Use the function `geom_col()`, map percentages and the values of the features to the position scales of the coordinate system, map the the value of the target to the `fill` aesthetic, and select colors to use with the function `scale_fill_manual()`. Label each slice of the bars with the percentage it represents

```
# Step 1: Construct a DataFrame containing the data to be plotted
```

```
# Step 2: Create the plot
```

Exercise 3: Logistic regression

Use the empty code cells below to do the following:

1. Fit a multiple logistic regression model of the target Churn on the features Subscription Length, Seconds of Use, Frequency of Use, Customer Value, Complains, and Active. Put the coefficients in a new DataFrame and output it. Compute the error rate, construct a confusion matrix, and compute precision, recall, and the F1 score. Finally, Plot the ROC curve and compute the AUC

2. Create a **separation plot**. A separation plot displays a vertical line segment for each unit of observation i, ordered by the estimated response probability $\widehat{p}(x_i)$. The vertical line segments are colored by the observed value of the target y_i. A line graph representing $\widehat{p}(x_i)$ is typical layered on top of the vertical line segments. Start by constructing a DataFrame containing y_i and $\widehat{p}(x_i)$. Order the DataFrame by the predicted values and create and index variable that keeps track of the order. Then create the plot using the functions `geom_segment()` and `geom_line()`. Map the index variable to the horizontal axis of the coordinate system and y_i to the `color` aesthetic. To learn more separation plots, have a look at this paper. You can access it through https://soeg.kb.dk/ using your WAYF account. Take a moment to think about what the plots and goodness of fit metrics tell you about the model and the data

```
# Fit a multiple linear regression model and output the coefficients
```

```
# Compute goodness of fit metrics and output them
```

```
# Plot the ROC curve and compute the AUC
```

```
# Create a separation plot
```

Exercise 4: Ridge regression

Use the empty code cells below to do the following:

1. Use the scikit-learn function `PolynomialFeatures()` to construct second-degree polynomial terms of all the quantitative features in the Iranian telecom data as well as all interactions among them. Collect this set of engineered features along with the qualitative features in the Iranian telecom data in a DataFrame. Then use the scikitlearn function `StandardScaler()` to standardize the features
2. Use ridge regression to estimate logistic models of the target Churn on the full feature set for 100 values of λ between 0.01 and 10. You can use the NumPy function np.geomspace to create a sequence of numbers that are evenly spaced on a log scale. Note that the argument `C` of the scikit-learn function `LogisticRegression()` lets you specify the quantity $1/\lambda$

3. Once you have estimated the models, plot the ridge coefficient path. To do this you will need to store the estimates all the ridge regression fits in a single column of a DataFrame the other columns of which map the estimates to features names and values of λ. Use color to highlight the 5 most predictive features as determined by largest estimates in absolute value for the highest value of λ used

```
# Construct the full standardized feature set
```

```
# Fit ridge regression models for 100 values of lambda
```

```
# Plot the ridge coefficient path
```

Exercise 5: The LASSO

Use the empty code cell below to do the following:

1. Use the LASSO to estimate logistic models of the target Churn on the full feature set that you created in exercise 6 for 200 values of λ between 0.01 and 100
2. Once you have estimated the models, plot the LASSO coefficient path. Take a moment to compare the LASSO coefficient path to the ridge coefficient path

```
# Fit the LASSO for 100 values of lambda
```

```
# Plot the LASSO coefficient path
```

Exercise 6: K-nearest neighbors for classification

Use the empty code cell to do the following:

1. Fit KNN classifiers of the target Churn on the the features Subscription Length, Seconds of Use, Frequency of Use, Customer Value, Complains, and Active for the following values of K: 1, 5, 10, 20, 40, 80, 160. Standardize the features before fitting the model
2. Create a new DataFrame in which the first column contains the value of K for each fit, the second column contains the error rate, the third column contains the precision score, the fourth column contains the recall score, the sixth column contains the F1 score, and seventh column contains the AUC. Output the DataFrame. Take a moment

to think about what the goodness of fit metrics tell you about the data and the models. For $K = 128$ you should get missing values for precision and the F1 score and a recall of zero. What do these numbers tell you?

```
# Fit KNN classifiers and compute GOF metrics
```

```
# Create a DataFrame containing K and GOF metrics and output it
```