DefineAndSolveMLProblem

July 29, 2025

1 Lab 8: Define and Solve an ML Problem of Your Choosing

[1]:
```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
```

In this lab assignment, you will follow the machine learning life cycle and implement a model to solve a machine learning problem of your choosing. You will select a data set and choose a predictive problem that the data set supports. You will then inspect the data with your problem in mind and begin to formulate a project plan. You will then implement the machine learning project plan.

You will complete the following tasks:

1. Build Your DataFrame
2. Define Your ML Problem
3. Perform exploratory data analysis to understand your data.
4. Define Your Project Plan
5. Implement Your Project Plan:
    - Prepare your data for your model.
    - Fit your model to the training data and evaluate your model.
    - Improve your model's performance.

1.1 Part 1: Build Your DataFrame

You will have the option to choose one of four data sets that you have worked with in this program:

- The "census" data set that contains Census information from 1994: `censusData.csv`
- Airbnb NYC "listings" data set: `airbnbListingsData.csv`
- World Happiness Report (WHR) data set: `WHR2018Chapter2OnlineData.csv`
- Book Review data set: `bookReviewsData.csv`

Note that these are variations of the data sets that you have worked with in this program. For example, some do not include some of the preprocessing necessary for specific models.

Load a Data Set and Save it as a Pandas DataFrame The code cell below contains filenames (path + filename) for each of the four data sets available to you.

Task: In the code cell below, use the same method you have been using to load the data using `pd.read_csv()` and save it to DataFrame `df`.

You can load each file as a new DataFrame to inspect the data before choosing your data set.

[2]:
```
# File names of the four data sets
adultDataSet_filename = os.path.join(os.getcwd(), "data", "censusData.csv")
airbnbDataSet_filename = os.path.join(os.getcwd(), "data", "airbnbListingsData.
 ↪csv")
WHRDataSet_filename = os.path.join(os.getcwd(), "data",␣
 ↪"WHR2018Chapter2OnlineData.csv")
bookReviewDataSet_filename = os.path.join(os.getcwd(), "data", "bookReviewsData.
 ↪csv")


df = pd.read_csv(WHRDataSet_filename)
df.head()

print(df.columns)
```

```
Index(['country', 'year', 'Life Ladder', 'Log GDP per capita',
       'Social support', 'Healthy life expectancy at birth',
       'Freedom to make life choices', 'Generosity',
       'Perceptions of corruption', 'Positive affect', 'Negative affect',
       'Confidence in national government', 'Democratic Quality',
       'Delivery Quality', 'Standard deviation of ladder by country-year',
       'Standard deviation/Mean of ladder by country-year',
       'GINI index (World Bank estimate)',
       'GINI index (World Bank estimate), average 2000-15',
       'gini of household income reported in Gallup, by wp5-year'],
      dtype='object')
```

1.2 Part 2: Define Your ML Problem

Next you will formulate your ML Problem. In the markdown cell below, answer the following questions:

1. List the data set you have chosen.
2. What will you be predicting? What is the label?
3. Is this a supervised or unsupervised learning problem? Is this a clustering, classification or regression problem? Is it a binary classificaiton or multi-class classifiction problem?
4. What are your features? (note: this list may change after your explore your data)
5. Explain why this is an important problem. In other words, how would a company create value with a model that predicts this label?

1. The data set we have chosen is the World Happiness Report
2. We will be predicting healthy life expectancy at birth and that is also the label
3. This is a supervised linear regression problem. This is not a classification problem.

4. Our features will be features besides the label and any with >100 NaN values
5. This is an important problem to explore because it can help us understand where we need to allocate more of our health resources to increase the life expectancy in that particular country. If we know where the problem is, then we can better understand how to target and solve it.

1.3 Part 3: Understand Your Data

The next step is to perform exploratory data analysis. Inspect and analyze your data set with your machine learning problem in mind. Consider the following as you inspect your data:

1. What data preparation techniques would you like to use? These data preparation techniques may include:
    - addressing missingness, such as replacing missing values with means
    - finding and replacing outliers
    - renaming features and labels
    - finding and replacing outliers
    - performing feature engineering techniques such as one-hot encoding on categorical features
    - selecting appropriate features and removing irrelevant features
    - performing specific data cleaning and preprocessing techniques for an NLP problem
    - addressing class imbalance in your data sample to promote fair AI
2. What machine learning model (or models) you would like to use that is suitable for your predictive problem and data?
    - Are there other data preparation techniques that you will need to apply to build a balanced modeling data set for your problem and model? For example, will you need to scale your data?
3. How will you evaluate and improve the model's performance?
    - Are there specific evaluation metrics and methods that are appropriate for your model?

Think of the different techniques you have used to inspect and analyze your data in this course. These include using Pandas to apply data filters, using the Pandas `describe()` method to get insight into key statistics for each column, using the Pandas `dtypes` property to inspect the data type of each column, and using Matplotlib and Seaborn to detect outliers and visualize relationships between features and labels. If you are working on a classification problem, use techniques you have learned to determine if there is class imbalance.

Task: Use the techniques you have learned in this course to inspect and analyze your data. You can import additional packages that you have used in this course that you will need to perform this task.

Note: You can add code cells if needed by going to the Insert menu and clicking on Insert Cell Below in the drop-drown menu.

[4]:

```
#1. We will address missing values by removing them, as well as finding and␣
 ↪replacing outliers. We will also combine data from the same countries to␣
 ↪make it more efficient. We will remove irrelevant features.
#2. We will use a logistic regression model to analyze our data. We will graph␣
 ↪the difference between our testing and training results to make sure that it␣
 ↪isn't overfitting. We will also change hyperparameters to see which works␣
 ↪best.
```

1.4 Part 4: Define Your Project Plan

Now that you understand your data, in the markdown cell below, define your plan to implement the remaining phases of the machine learning life cycle (data preparation, modeling, evaluation) to solve your ML problem. Answer the following questions:

- Do you have a new feature list? If so, what are the features that you chose to keep and remove after inspecting the data?
- Explain different data preparation techniques that you will use to prepare your data for modeling.
- What is your model (or models)?
- Describe your plan to train your model, analyze its performance and then improve the model. That is, describe your model building, validation and selection plan to produce a model that generalizes well to new data.

1. We plan to see how many values are missing when cleaning our data and decide our top 5 features from there. Our model is a logistic regression model. We plan to train our model and

1.5 Part 5: Implement Your Project Plan

Task: In the code cell below, import additional packages that you have used in this course that you will need to implement your project plan.

[5]:
```
import os
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪precision_recall_curve
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import cross_val_score
```

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
```

Task: Use the rest of this notebook to carry out your project plan.

You will:

1. Prepare your data for your model.
2. Fit your model to the training data and evaluate your model.
3. Improve your model's performance by performing model selection and/or feature selection techniques to find best model for your problem.

Add code cells below and populate the notebook with commentary, code, analyses, results, and figures as you see fit.

[6]:
```
#Making a feature list with all the features
feature_list = df.columns.tolist()
feature_list.remove('Healthy life expectancy at birth')

#HANDLING MISSING DATA
#Determining which features have >100 NaN values
nan_count = np.sum(df.isnull(), axis = 0)
nan_count

#Removing those features
missing = df.isnull().sum()
threshold = 100
clean_features = [col for col in feature_list if missing[col] <= threshold]
df_cleaned = df[clean_features]
clean_features

#Creating new columns
df['GDP_na'] = df['Log GDP per capita'].isnull()
df['Social_support_na'] = df['Social support'].isnull()
df['Life_choices_na'] = df['Freedom to make life choices'].isnull()
df['Generosity_na'] = df['Generosity'].isnull()
df['Corruption_na'] = df['Perceptions of corruption'].isnull()
df['Positive_affect_na'] = df['Positive affect'].isnull()
df['Negative_affect_na'] = df['Negative affect'].isnull()

#Fill in with mean values
mean_GDP = df['Log GDP per capita'].mean()
df['Log GDP per capita'] = df['Log GDP per capita'].fillna(mean_GDP)

mean_support = df['Social support'].mean()
df['Social support'] = df['Social support'].fillna(mean_support)

mean_lifechoices = df['Freedom to make life choices'].mean()
```

```
df['Freedom to make life choices'] = df['Freedom to make life choices'].
 ↪fillna(mean_lifechoices)

mean_generosity = df['Generosity'].mean()
df['Generosity'] = df['Generosity'].fillna(mean_generosity)

mean_perceptions = df['Perceptions of corruption'].mean()
df['Perceptions of corruption'] = df['Perceptions of corruption'].
 ↪fillna(mean_perceptions)

mean_positive = df['Positive affect'].mean()
df['Positive affect'] = df['Positive affect'].fillna(mean_positive)

mean_negative = df['Negative affect'].mean()
df['Negative affect'] = df['Negative affect'].fillna(mean_negative)

#Removing year column and averaging values of countries through the years
if 'year' in df.columns:
    df = df.drop(columns=['year'], errors='ignore')

df_avg = df. groupby ('country'). mean (numeric_only=True). reset_index()
print(df_avg.head () )
print(df. head ( ))

#Dropping unneeded columns
cols_to_drop = [
    'GINI index (World Bank estimate)',
    'GINI index (World Bank estimate), average 2000-15',
    'Confidence in national government',
    'gini of household income reported in Gallup, by wp5-year',
    'Delivery Quality',
    'Democratic Quality',
]
df_avg = df_avg.drop(columns=cols_to_drop, errors='ignore')


#DEFINING FEATURES AND LABEL

label = 'Healthy life expectancy at birth'

#Remove country column
df_model = df_avg.drop(columns=['country'], errors='ignore')

#Drop rows where target is NaN
df_model = df_model.dropna(subset=[label])

#Define X and y
```

```
X = df_model.drop(columns=[label])
y = df_model[label]

#Outlier removal

#Fit your best model on the full dataset
model = Ridge(alpha=grid_search.best_params_['alpha'])
model.fit(X, y)

#Predict on the full dataset
y_pred_all = model.predict(X)

#Calculate residuals and standardized residuals
residuals = y - y_pred_all
std_residuals = (residuals - np.mean(residuals)) / np.std(residuals)

#Set a threshold for outliers
threshold = 3
outlier_mask = np.abs(std_residuals) > threshold

#Report outliers
print(f"Number of outliers: {np.sum(outlier_mask)}")

#Filter them out
X_no_outliers = X[~outlier_mask]
y_no_outliers = y[~outlier_mask]


#Splitting in train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X_no_outliers, y_no_outliers, test_size=0.25, random_state=123)

#TRAINING LINEAR REGRESSION MODEL
model = LinearRegression()
model.fit(X_train, y_train)

#EVALUATE THE MODEL
from sklearn.metrics import mean_squared_error, r2_score

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"MSE: {mse:.2f}")
print(f"R² Score: {r2:.2f}")

#Get predictions
```

```
y_train_pred = model.predict(X_train)
y_test_pred = model.predict(X_test)

#Plotting
plt.figure(figsize=(10,5))

#Training set plot
plt.subplot(1, 2, 1)
plt.scatter(y_train, y_train_pred, alpha=0.7, color='blue')
plt.plot([y_train.min(), y_train.max()], [y_train.min(), y_train.max()], 'r--')
plt.xlabel('Actual (Train)')
plt.ylabel('Predicted (Train)')
plt.title('Training Set: Actual vs Predicted')
plt.grid(True)

#Test set plot
plt.subplot(1, 2, 2)
plt.scatter(y_test, y_test_pred, alpha=0.7, color='green')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.xlabel('Actual (Test)')
plt.ylabel('Predicted (Test)')
plt.title('Test Set: Actual vs Predicted')
plt.grid(True)

plt.tight_layout()
plt.show()

scores = cross_val_score(model, X, y, cv=5, scoring='r2')
print(f"Average CV R² score: {scores.mean():.4f}")

#Performing grid search
#Define ridge model
ridge = Ridge(random_state=123)

#Define the hyperparameter grid to search
param_grid = {
    'alpha': [0.01, 0.1, 1, 10, 100, 200]
}

#GridSearchCV with 5-fold cross-validation and R² scoring
grid_search = GridSearchCV(
    estimator=ridge,
    param_grid=param_grid,
    cv=5,
    scoring='r2',
    n_jobs=-1,
    verbose=1
```

```
)

# Fit grid search on training data
grid_search.fit(X_train, y_train)

#View best parameters and best score
print("Best alpha:", grid_search.best_params_['alpha'])
print(f"Best cross-validated R²: {grid_search.best_score_:.4f}")

#Use the best estimator to predict and evaluate on test data
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Test MSE: {mse:.4f}")
print(f"Test R²: {r2:.4f}")

#Get feature names and coefficients from the best Ridge model
feature_names = X.columns
coefs = best_model.coef_

#Create a DataFrame for better visualization
coef_df = pd.DataFrame({
    'Feature': feature_names,
    'Coefficient': coefs
})


#Sort by absolute value of coefficients
coef_df['Abs_Coefficient'] = np.abs(coef_df['Coefficient'])
coef_df_sorted = coef_df.sort_values(by='Abs_Coefficient', ascending=True)

#Plot
plt.figure(figsize=(10, 8))
plt.barh(coef_df_sorted['Feature'], coef_df_sorted['Coefficient'],␣
 ↪color='skyblue')
plt.axvline(0, color='gray', linestyle='--')
plt.title('Feature Importance (Coefficient Values from Ridge Regression)')
plt.xlabel('Coefficient Value')
plt.ylabel('Feature')
plt.tight_layout()
plt.show()

#CLEAN THE DATA MORE
#Remove features where coefficient is exactly zero
features_to_keep = coef_df[coef_df['Coefficient'] != 0]['Feature'].tolist()
```

```
print("features to keep: ")
print(features_to_keep)

#Filter dataset to only keep those features
X_filtered = X[features_to_keep]

X_train_filtered, X_test_filtered, y_train, y_test =␣
 ↪train_test_split(X_filtered, y, test_size=0.2, random_state=123)

model = Ridge(alpha=grid_search.best_params_['alpha'])
model.fit(X_train_filtered, y_train)

#Evaluate the filtered model
y_pred = model.predict(X_test_filtered)
print("Filtered Test R²:", r2_score(y_test, y_pred))

#Convert column lists to sets
original_features = set(X.columns)
filtered_features = set(X_filtered.columns)

#Find dropped columns
dropped_features = original_features - filtered_features

#Print the results
if dropped_features:
    print("Dropped features:")
    for feature in dropped_features:
        print("-", feature)
else:
    print("No features were dropped.")
```

```
       country  Life Ladder  Log GDP per capita  Social support  \
0  Afghanistan     3.806614            7.419697        0.517146
1      Albania     4.988791            9.247059        0.723204
2      Algeria     5.555004            9.501728        0.805639
3       Angola     4.420299            8.713935        0.737973
4    Argentina     6.406131            9.826051        0.906080

   Healthy life expectancy at birth  Freedom to make life choices  Generosity  \
0                         50.838271                      0.544895    0.118428
1                         68.027213                      0.626155   -0.105019
2                         64.984461                      0.600590   -0.138797
3                         51.729801                      0.455957   -0.077940
4                         66.764205                      0.753122   -0.154544

   Perceptions of corruption  Positive affect  Negative affect  …  \
0                   0.826794         0.580873         0.301283  …
```

```
1                   0.859691         0.642628         0.303256  …
2                   0.692192         0.631932         0.265079  …
3                   0.867018         0.613339         0.351173  …
4                   0.844038         0.840998         0.273187  …

   GINI index (World Bank estimate)  \
0                               NaN
1                          0.290000
2                          0.276000
3                               NaN
4                          0.447667

   GINI index (World Bank estimate), average 2000-15  \
0                                                NaN
1                                           0.303250
2                                           0.276000
3                                           0.427000
4                                           0.476067

   gini of household income reported in Gallup, by wp5-year  GDP_na  \
0                                           0.385668            0.0
1                                           0.492998            0.0
2                                           0.491331            0.0
3                                           0.514382            0.0
4                                           0.349463            0.0

   Social_support_na  Life_choices_na  Generosity_na  Corruption_na  \
0           0.000000         0.000000       0.000000       0.000000
1           0.000000         0.000000       0.000000       0.000000
2           0.166667         0.333333       0.333333       0.333333
3           0.000000         0.000000       0.000000       0.000000
4           0.000000         0.000000       0.000000       0.000000

   Positive_affect_na  Negative_affect_na
0            0.000000            0.000000
1            0.000000            0.000000
2            0.166667            0.166667
3            0.000000            0.000000
4            0.000000            0.000000

[5 rows x 25 columns]
       country  Life Ladder  Log GDP per capita  Social support  \
0  Afghanistan     3.723590            7.168690        0.450662
1  Afghanistan     4.401778            7.333790        0.552308
2  Afghanistan     4.758381            7.386629        0.539075
3  Afghanistan     3.831719            7.415019        0.521104
4  Afghanistan     3.782938            7.517126        0.520637
```

```
   Healthy life expectancy at birth  Freedom to make life choices  Generosity  \
0                         49.209663                      0.718114    0.181819
1                         49.624432                      0.678896    0.203614
2                         50.008961                      0.600127    0.137630
3                         50.367298                      0.495901    0.175329
4                         50.709263                      0.530935    0.247159

   Perceptions of corruption  Positive affect  Negative affect  …  \
0                   0.881686         0.517637         0.258195  …
1                   0.850035         0.583926         0.237092  …
2                   0.706766         0.618265         0.275324  …
3                   0.731109         0.611387         0.267175  …
4                   0.775620         0.710385         0.267919  …

   GINI index (World Bank estimate)  \
0                               NaN
1                               NaN
2                               NaN
3                               NaN
4                               NaN

   GINI index (World Bank estimate), average 2000-15  \
0                                                NaN
1                                                NaN
2                                                NaN
3                                                NaN
4                                                NaN

   gini of household income reported in Gallup, by wp5-year  GDP_na  \
0                                                       NaN   False
1                                                  0.441906   False
2                                                  0.327318   False
3                                                  0.336764   False
4                                                  0.344540   False

   Social_support_na  Life_choices_na  Generosity_na  Corruption_na  \
0              False            False          False          False
1              False            False          False          False
2              False            False          False          False
3              False            False          False          False
4              False            False          False          False

   Positive_affect_na  Negative_affect_na
0               False               False
1               False               False
2               False               False
3               False               False
4               False               False
```

```
[5 rows x 25 columns]
```

```
␣
↪---------------------------------------------------------------------------

NameError                                 Traceback (most recent call␣
↪last)

Cell In[6], line 85
 80 y = df_model[label]
 82 #Outlier removal
 83
 84 #Fit your best model on the full dataset
---> 85 model = Ridge(alpha=grid_search.best_params_['alpha'])
 86 model.fit(X, y)
 88 #Predict on the full dataset


NameError: name 'grid_search' is not defined
```

[ ]:

[ ]: