ImplementMLProjectPlan

August 11, 2023

1 Lab 8: Implement Your Machine Learning Project Plan

In this lab assignment, you will implement the machine learning project plan you created in the written assignment. You will:

1. Load your data set and save it to a Pandas DataFrame.
2. Perform exploratory data analysis on your data to determine which feature engineering and data preparation techniques you will use.
3. Prepare your data for your model and create features and a label.
4. Fit your model to the training data and evaluate your model.
5. Improve your model by performing model selection and/or feature selection techniques to find best model for your problem.

1.0.1 Import Packages

Before you get started, import a few packages.

[135]:
```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
```

Task: In the code cell below, import additional packages that you have used in this course that you will need for this task.

[244]:
```
# YOUR CODE HERE
from scipy.stats.mstats import winsorize
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score, confusion_matrix,␣
↪precision_recall_curve
from sklearn.metrics import r2_score
```

1.1 Part 1: Load the Data Set

You have chosen to work with one of four data sets. The data sets are located in a folder named "data." The file names of the three data sets are as follows:

- The "adult" data set that contains Census information from 1994 is located in file `adultData.csv`
- The airbnb NYC "listings" data set is located in file `airbnbListingsData.csv`
- The World Happiness Report (WHR) data set is located in file `WHR2018Chapter2OnlineData.csv`
- The book review data set is located in file `bookReviewsData.csv`

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

[245]:
```
# YOUR CODE HERE
WHRDataSet_filename = os.path.join(os.getcwd(), "data",␣
↪"WHR2018Chapter2OnlineData.csv")
df = pd.read_csv(WHRDataSet_filename, header=0)
```

1.2 Part 2: Exploratory Data Analysis

The next step is to inspect and analyze your data set with your machine learning problem and project plan in mind.

This step will help you determine data preparation and feature engineering techniques you will need to apply to your data to build a balanced modeling data set for your problem and model. These data preparation techniques may include: * addressing missingness, such as replacing missing values with means * renaming features and labels * finding and replacing outliers * performing winsorization if needed * performing one-hot encoding on categorical features * performing vectorization for an NLP problem * addressing class imbalance in your data sample to promote fair AI

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.

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.

[246]:
```
# YOUR CODE HERE
df.describe()
```

[246]:
```
              year  Life Ladder  Log GDP per capita  Social support  \
count  1562.000000  1562.000000         1535.000000     1549.000000
mean   2011.820743     5.433676            9.220822        0.810669
std       3.419787     1.121017            1.184035        0.119370
min    2005.000000     2.661718            6.377396        0.290184
25%    2009.000000     4.606351            8.310665        0.748304
```

```
50%    2012.000000    5.332600           9.398610        0.833047
75%    2015.000000    6.271025          10.190634        0.904329
max    2017.000000    8.018934          11.770276        0.987343

       Healthy life expectancy at birth  Freedom to make life choices  \
count                       1553.000000                   1533.000000
mean                          62.249887                      0.728975
std                            7.960671                      0.145408
min                           37.766476                      0.257534
25%                           57.299580                      0.633754
50%                           63.803192                      0.748014
75%                           68.098228                      0.843628
max                           76.536362                      0.985178

        Generosity  Perceptions of corruption  Positive affect  \
count  1482.000000                1472.000000      1544.000000
mean      0.000079                   0.753622         0.708969
std       0.164202                   0.185538         0.107644
min      -0.322952                   0.035198         0.362498
25%      -0.114313                   0.697359         0.621471
50%      -0.022638                   0.808115         0.717398
75%       0.094649                   0.880089         0.800858
max       0.677773                   0.983276         0.943621

       Negative affect  Confidence in national government  Democratic Quality  \
count      1550.000000                        1401.000000         1391.000000
mean          0.263171                           0.480207           -0.126617
std           0.084006                           0.190724            0.873259
min           0.083426                           0.068769           -2.448228
25%           0.204116                           0.334732           -0.772010
50%           0.251798                           0.463137           -0.225939
75%           0.311515                           0.610723            0.665944
max           0.704590                           0.993604            1.540097

       Delivery Quality  Standard deviation of ladder by country-year  \
count       1391.000000                                   1562.000000
mean           0.004947                                      2.003501
std            0.981052                                      0.379684
min           -2.144974                                      0.863034
25%           -0.717463                                      1.737934
50%           -0.210142                                      1.960345
75%            0.717996                                      2.215920
max            2.184725                                      3.527820

       Standard deviation/Mean of ladder by country-year  \
count                                        1562.000000
mean                                            0.387271
```

```
std                                                     0.119007
min                                                     0.133908
25%                                                     0.309722
50%                                                     0.369751
75%                                                     0.451833
max                                                     1.022769

       GINI index (World Bank estimate)  \
count                        583.000000
mean                           0.372846
std                            0.086609
min                            0.241000
25%                            0.307000
50%                            0.349000
75%                            0.433500
max                            0.648000

       GINI index (World Bank estimate), average 2000-15  \
count                                          1386.000000
mean                                              0.386948
std                                               0.083694
min                                               0.228833
25%                                               0.321583
50%                                               0.371000
75%                                               0.433104
max                                               0.626000

       gini of household income reported in Gallup, by wp5-year
count                                          1205.000000
mean                                              0.445204
std                                               0.105410
min                                               0.223470
25%                                               0.368531
50%                                               0.425395
75%                                               0.508579
max                                               0.961435
```

[247]:
```
df.dtypes
```

[247]:
```
country                                                     object
year                                                         int64
Life Ladder                                                float64
Log GDP per capita                                         float64
Social support                                             float64
Healthy life expectancy at birth                           float64
Freedom to make life choices                               float64
Generosity                                                 float64
Perceptions of corruption                                  float64
```

```
Positive affect                                          float64
Negative affect                                          float64
Confidence in national government                        float64
Democratic Quality                                       float64
Delivery Quality                                         float64
Standard deviation of ladder by country-year             float64
Standard deviation/Mean of ladder by country-year        float64
GINI index (World Bank estimate)                         float64
GINI index (World Bank estimate), average 2000-15        float64
gini of household income reported in Gallup, by wp5-year    float64
dtype: object
```

1.2.1 Check the data for any missingness.

[248]:
```python
nan_count = np.sum(df.isnull(), axis = 0)
nan_count
```

[248]:
```
country                                                     0
year                                                        0
Life Ladder                                                 0
Log GDP per capita                                         27
Social support                                             13
Healthy life expectancy at birth                            9
Freedom to make life choices                               29
Generosity                                                 80
Perceptions of corruption                                  90
Positive affect                                            18
Negative affect                                            12
Confidence in national government                         161
Democratic Quality                                        171
Delivery Quality                                          171
Standard deviation of ladder by country-year                0
Standard deviation/Mean of ladder by country-year           0
GINI index (World Bank estimate)                          979
GINI index (World Bank estimate), average 2000-15         176
gini of household income reported in Gallup, by wp5-year  357
dtype: int64
```

We'll also look for columns with high correlation values to narrow down the columns used in training.

[249]:
```python
df.corr()['Life Ladder']
```

[249]:
```
year                                                     -0.014505
Life Ladder                                               1.000000
Log GDP per capita                                        0.779476
Social support                                            0.700299
Healthy life expectancy at birth                          0.729852
Freedom to make life choices                              0.526058
```

```
Generosity                                                 0.204910
Perceptions of corruption                                 -0.425013
Positive affect                                            0.554462
Negative affect                                           -0.267492
Confidence in national government                         -0.085543
Democratic Quality                                         0.607034
Delivery Quality                                           0.706673
Standard deviation of ladder by country-year              -0.154257
Standard deviation/Mean of ladder by country-year         -0.756076
GINI index (World Bank estimate)                          -0.097255
GINI index (World Bank estimate), average 2000-15         -0.172745
gini of household income reported in Gallup, by wp5-year  -0.294080
Name: Life Ladder, dtype: float64
```

[250]:

```
exclude = ['Life Ladder']
corrs = df.corr()['Life Ladder'].drop(exclude, axis = 0)
corrs_sorted = corrs.sort_values(ascending=False)
print(corrs_sorted)
```

```
Log GDP per capita                                         0.779476
Healthy life expectancy at birth                           0.729852
Delivery Quality                                           0.706673
Social support                                             0.700299
Democratic Quality                                         0.607034
Positive affect                                            0.554462
Freedom to make life choices                               0.526058
Generosity                                                 0.204910
year                                                      -0.014505
Confidence in national government                         -0.085543
GINI index (World Bank estimate)                          -0.097255
Standard deviation of ladder by country-year              -0.154257
GINI index (World Bank estimate), average 2000-15         -0.172745
Negative affect                                           -0.267492
gini of household income reported in Gallup, by wp5-year  -0.294080
Perceptions of corruption                                 -0.425013
Standard deviation/Mean of ladder by country-year         -0.756076
Name: Life Ladder, dtype: float64
```

To my surprise, GINI index, a measure of income inequality between the lowest and highest earners, was negatively correlated with the label, "Life Ladder". This may be a consequence of relatively high missingness found earlier. For my model, I will include the positively correlated columns as my features.

[251]:

```
#Create a list of columns that will be used to create a new dataframe
newFeatures = list(corrs_sorted[corrs_sorted > 0].index)
newFeatures.append("Life Ladder")
```

I will be including year despite a slightly negative correlation, as functions in scientific inquiries are often functions of time, where we may expect life ladder to improve over time. I will be excluding "Country" as a feature because using one-hot encoding would only allow for about 15 examples per feature, and bloat the number of features to nearly 200, which does not work well for decision trees.

[252]:
```
#Create a new dataframe using the features that I deemed most relevant.
df_new = df[newFeatures]
df_new.head()
```

[252]:
```
   Log GDP per capita  Healthy life expectancy at birth  Delivery Quality  \
0            7.168690                         49.209663         -1.655084
1            7.333790                         49.624432         -1.635025
2            7.386629                         50.008961         -1.617176
3            7.415019                         50.367298         -1.616221
4            7.517126                         50.709263         -1.404078

   Social support  Democratic Quality  Positive affect  \
0        0.450662           -1.929690         0.517637
1        0.552308           -2.044093         0.583926
2        0.539075           -1.991810         0.618265
3        0.521104           -1.919018         0.611387
4        0.520637           -1.842996         0.710385

   Freedom to make life choices  Generosity  Life Ladder
0                      0.718114    0.181819     3.723590
1                      0.678896    0.203614     4.401778
2                      0.600127    0.137630     4.758381
3                      0.495901    0.175329     3.831719
4                      0.530935    0.247159     3.782938
```

[253]:
```
#Find the new nan count for this data
nan_count = np.sum(df_new.isnull(), axis = 0)
nan_count_series = pd.Series(nan_count, index=df_new.columns)
```

[254]:
```
for col_name, nan_count_value in nan_count_series.items():
    mean = df_new[col_name].mean()
    df_new.loc[:, col_name] = df_new.loc[:, col_name].fillna(mean)
```

```
/home/codio/.local/lib/python3.6/site-packages/pandas/core/indexing.py:1781:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self.obj[item_labels[indexer[info_axis]]] = value
```

1.2.2 Check nan_count after replacing all NaN values with mean values

[255]:
```
nan_count = np.sum(df_new.isnull(), axis = 0)
nan_count
```

[255]:
```
Log GDP per capita                  0
Healthy life expectancy at birth    0
Delivery Quality                    0
Social support                      0
Democratic Quality                  0
Positive affect                     0
Freedom to make life choices        0
Generosity                          0
Life Ladder                         0
dtype: int64
```

[256]:
```
limits = [0.01, 0.01]  # Limits for winsorization

for col in df_new.columns:
    df_new[col] = winsorize(df_new[col], limits=limits)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

1.3 Part 3: Implement Your Project Plan

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

1. Prepare your data for your model and create features and a label.
2. Fit your model to the training data and evaluate your model.
3. Improve your model 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.

[257]:
```
# YOUR CODE HERE
#Create the y label and X feature set
y = df_new['Life Ladder']
X = df_new.drop(columns = 'Life Ladder', axis=1)
#Create training data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.10,␣
 ↪random_state=1234)
```

1.3.1 Perform Decision Tree Model Selection

In the cell below we're going to create a model and use a gridsearch with a cv=5 to find the best performing hyperparameter for max_depth with varying max_depth and min_samples_leaf.

[258]:
```
# Create a range of hyperparameter values for 'max_depth'.
#Note these are the same values as those we used above
hyperparams_depth = list(range(1,16))

# Create a range of hyperparameter values for 'min_samples_leaf'.
hyperparams_leaf = [25*2**n for n in range(0,3)]

# Create parameter grid.
param_grid={'max_depth':hyperparams_depth, 'min_samples_leaf':hyperparams_leaf}
param_grid
```

[258]:
```
{'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15],
 'min_samples_leaf': [25, 50, 100]}
```

[259]:
```
# Create a range of hyperparameter values for 'max_depth'
hyperparams_depth = list(range(1, 16))  # Start from 1

# Create a range of hyperparameter values for 'min_samples_leaf'
hyperparams_leaf = [25*2**n for n in range(0, 3)]

# Create parameter grid
param_grid = {'max_depth': hyperparams_depth, 'min_samples_leaf':␣
 ↪hyperparams_leaf}

# Create a DecisionTreeRegressor instance
tree_regressor = DecisionTreeRegressor()

# Create GridSearchCV instance
grid_search = GridSearchCV(tree_regressor, param_grid, cv=5,␣
 ↪scoring='neg_mean_squared_error')

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

# Get the best parameters and best estimator from the grid search
best_params = grid_search.best_params_
best_estimator = grid_search.best_estimator_

# Print the best parameters
print("Best Parameters:", best_params)

# Print the best estimator
print("Best Estimator:", best_estimator)
```

```python
mse_values = [-mse for mse in grid_search.cv_results_['mean_test_score']]
```

```
Best Parameters: {'max_depth': 7, 'min_samples_leaf': 25}
Best Estimator: DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse',
max_depth=7,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=25, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
```

Create a model using the best max_depth hyperparameter.

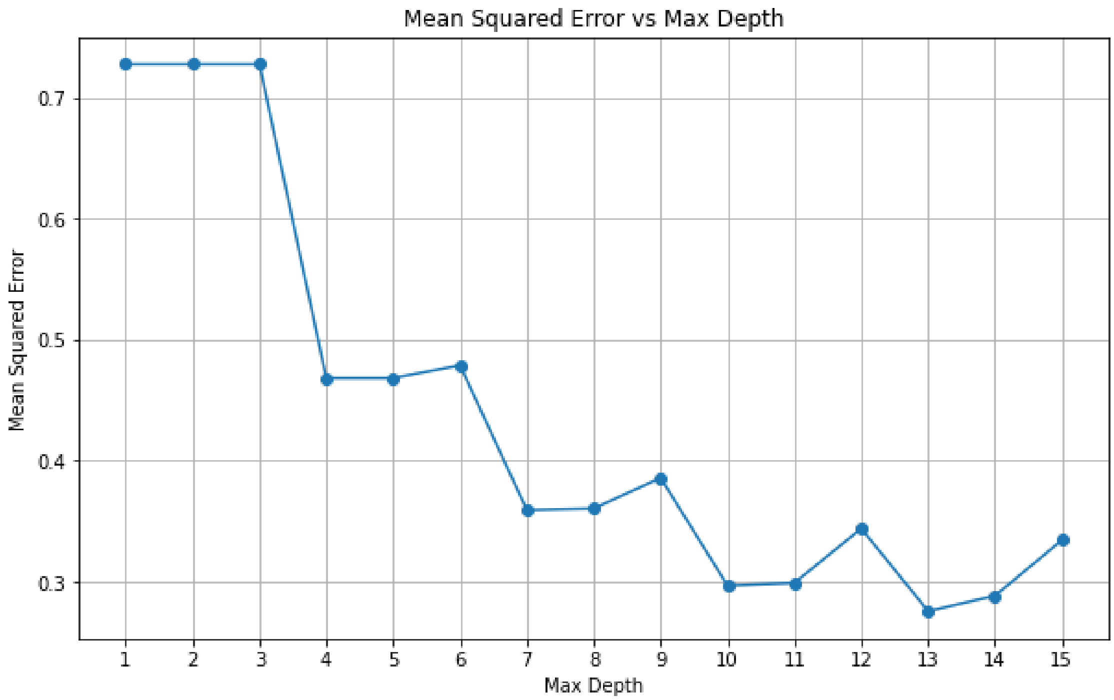
[260]:
```python
# 1. Create the model object below and assign to variable 'model_best'
model_best = DecisionTreeRegressor(**best_params)
# 2. Fit the model to the training data below
model_best.fit(X_train, y_train)

y_pred_model_best = model_best.predict(X_test)
```

1.3.2 Analyze the Mean Squared Error vs Max Depth

[261]:
```python
plt.figure(figsize=(10, 6))
plt.plot(hyperparams_depth, mse_values[:len(hyperparams_depth)], marker='o')

plt.title('Mean Squared Error vs Max Depth')
plt.xlabel('Max Depth')
plt.ylabel('Mean Squared Error')
plt.xticks(hyperparams_depth)
plt.grid(True)
plt.show()
```



[266]:
```
best_mse = -grid_search.best_score_
best_max_depth = grid_search.best_params_['max_depth']
print("Best Mean Squared Error:", best_mse)
print("Associated max_depth:", best_max_depth)
```

```
Best Mean Squared Error: 0.26486819808509054
Associated max_depth: 7
```

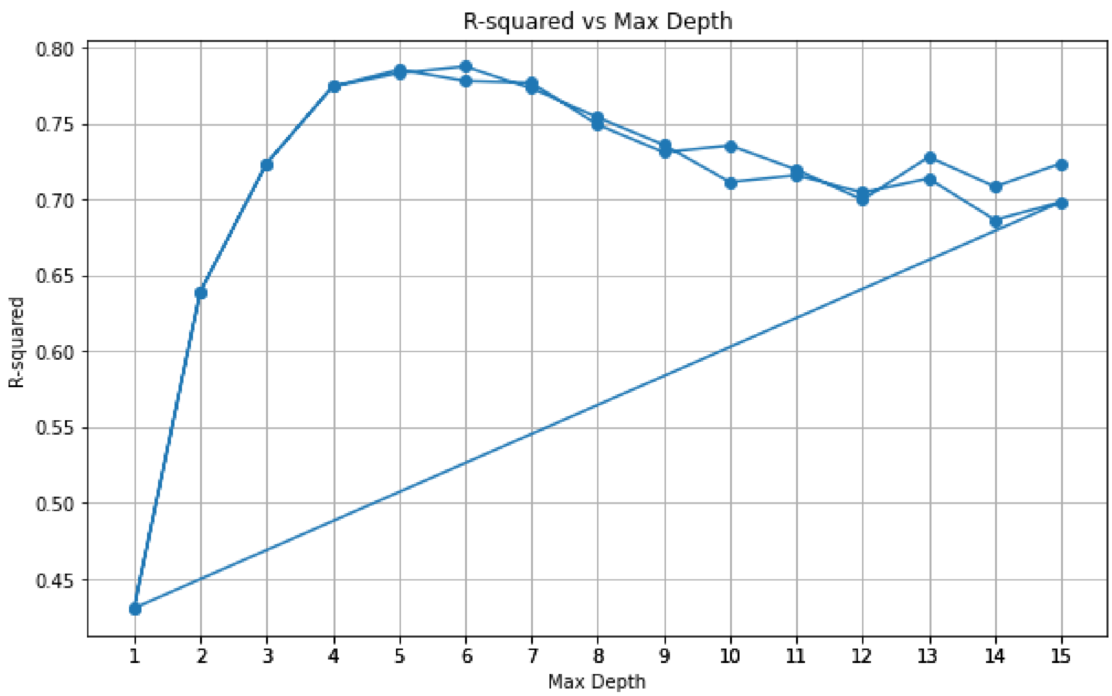
It appears that as max depth increases, MSE decreases. The best max_depth found through the gridsearch was 7 at 0.26486819808509054 MSE.

[263]:
```
# Calculate R-squared for each max_depth value
for max_depth in hyperparams_depth:
    tree_regressor = DecisionTreeRegressor(max_depth=max_depth)
    tree_regressor.fit(X_train, y_train)
    y_pred = tree_regressor.predict(X_test)
    r2 = r2_score(y_test, y_pred)
    max_depth_values.append(max_depth)
    r2_values.append(r2)

# Create a plot
plt.figure(figsize=(10, 6))
plt.plot(max_depth_values, r2_values, marker='o')

plt.title('R-squared vs Max Depth')
plt.xlabel('Max Depth')
```

```
plt.ylabel('R-squared')
plt.xticks(max_depth_values)
plt.grid(True)
plt.show()
```



According to my analysis, using a max depth of 7 maintains a relatively high predictive power at an R-squared value of 0.7632144685799015. Trying to balance a relatively low MSE value with high R-squared value to create the best model for this data with the highest predictive power is the goal of this exercise. I would recommend using the best_model at max_depth=7 in an attempt to balance R-squared and MSE.

[ ]:

```

```