ImplementMLProjectPlan

August 13, 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.

[32]:
```
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.

[33]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_squared_error
from scipy.stats.mstats import winsorize
```

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`.

[34]:
```
airbnbDataSet_filename = os.path.join(os.getcwd(), "data", "airbnbListingsData.
↪csv")
df = pd.read_csv(airbnbDataSet_filename)

df.head()
```

[34]:
```
                                                name  \
0                                Skylit Midtown Castle
1  Whole flr w/private bdrm, bath & kitchen(pls r...
2           Spacious Brooklyn Duplex, Patio + Garden
3                    Large Furnished Room Near B'way
4                       Cozy Clean Guest Room - Family Apt

                                         description  \
0  Beautiful, spacious skylit studio in the heart...
1  Enjoy 500 s.f. top floor in 1899 brownstone, w...
2  We welcome you to stay in our lovely 2 br dupl...
3  Please dont expect the luxury here just a bas...
4  Our best guests are seeking a safe, clean, spa...

                               neighborhood_overview    host_name  \
0  Centrally located in the heart of Manhattan ju...     Jennifer
1  Just the right mix of urban center and local n...  LisaRoxanne
2                                                NaN      Rebecca
3    Theater district, many restaurants around here.     Shunichi
4  Our neighborhood is full of restaurants and ca...    MaryEllen

                   host_location  \
0  New York, New York, United States
1  New York, New York, United States
2  Brooklyn, New York, United States
3  New York, New York, United States
4  New York, New York, United States

                                          host_about  host_response_rate  \
0  A New Yorker since 2000! My passion is creatin...                0.80
1  Laid-back Native New Yorker (formerly bi-coast...                0.09
2  Rebecca is an artist/designer, and Henoch is i...                1.00
3  I used to work for a financial industry but no...                1.00
```

```
4  Welcome to family life with my oldest two away...                 NaN

   host_acceptance_rate  host_is_superhost  host_listings_count  ...  \
0                  0.17               True                  8.0  ...
1                  0.69               True                  1.0  ...
2                  0.25               True                  1.0  ...
3                  1.00               True                  1.0  ...
4                   NaN               True                  1.0  ...

   review_scores_communication  review_scores_location  review_scores_value  \
0                         4.79                    4.86                 4.41
1                         4.80                    4.71                 4.64
2                         5.00                    4.50                 5.00
3                         4.42                    4.87                 4.36
4                         4.95                    4.94                 4.92

  instant_bookable calculated_host_listings_count  \
0            False                              3
1            False                              1
2            False                              1
3            False                              1
4            False                              1

   calculated_host_listings_count_entire_homes  \
0                                            3
1                                            1
2                                            1
3                                            0
4                                            0

   calculated_host_listings_count_private_rooms  \
0                                             0
1                                             0
2                                             0
3                                             1
4                                             1

   calculated_host_listings_count_shared_rooms  reviews_per_month  \
0                                            0               0.33
1                                            0               4.86
2                                            0               0.02
3                                            0               3.68
4                                            0               0.87

  n_host_verifications
0                    9
1                    6
```

```
2                    3
3                    4
4                    7

[5 rows x 50 columns]
```

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.

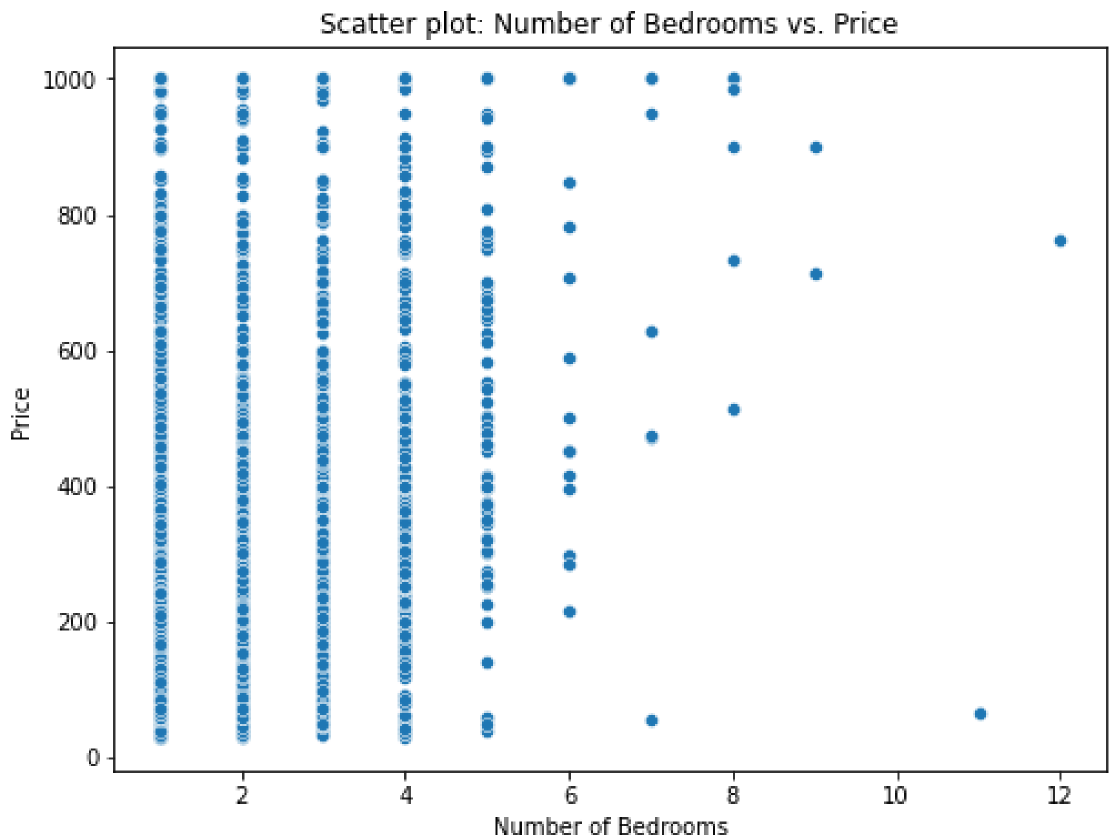
Note: You can add code cells if needed by going to the Insert menu and clicking on Insert Cell Below in the drop-drown menu.

[35]:
```
#Look at df stats
df.describe()

#Look at data types per column
types = df.dtypes
types

#Class imbalance
df['price'].value_counts()
```

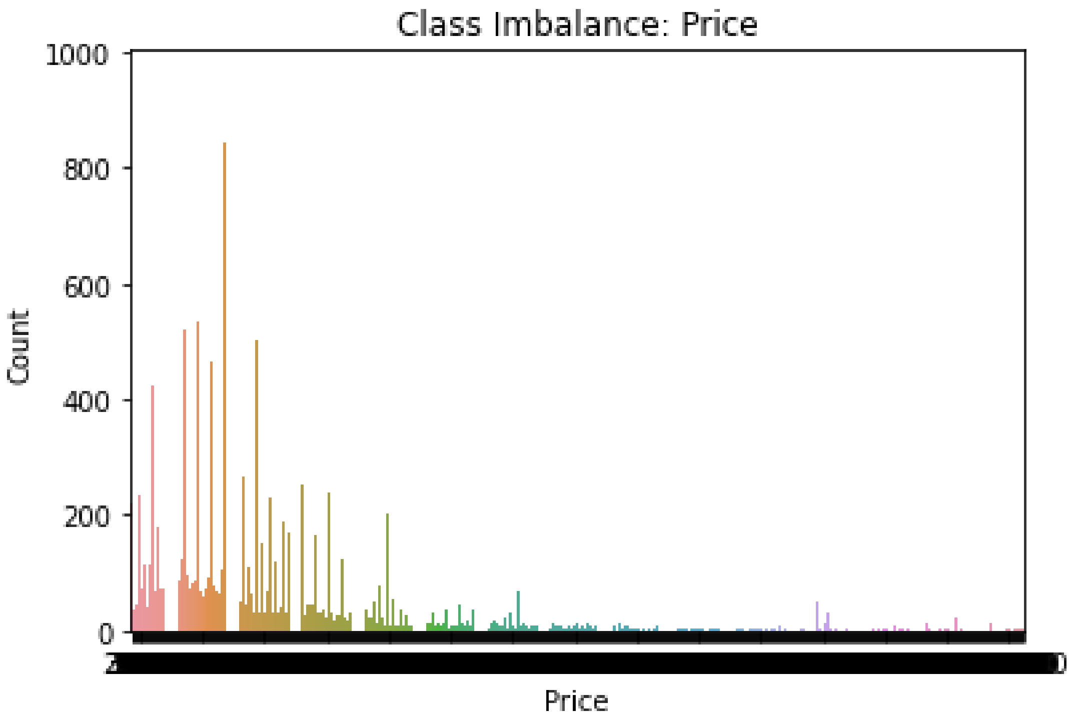
[35]:
```
150.0    955
100.0    844
60.0     650
50.0     627
75.0     623
        ...
287.0      1
815.0      1
609.0      1
468.0      1
985.0      1
```

```
Name: price, Length: 684, dtype: int64
```

[36]:
```
#Visualize number_of_bedrooms and price relationship via scatterplot
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='bedrooms', y='price')
plt.title("Scatter plot: Number of Bedrooms vs. Price")
plt.xlabel("Number of Bedrooms")
plt.ylabel("Price")
plt.show()
```



[26]:
```
#Visualize class imbalance using seaborn countplot
plt.figure(figsize=(6, 4))
sns.countplot(data=df, x='price')
plt.title("Class Imbalance: Price")
plt.xlabel("Price")
plt.ylabel("Count")
plt.show()
```



[37]:
```
#We can handle the missing values.
#I will convert all data variables to numerical values for flowing data
df.fillna(df.mean(), inplace=True)
#Above code replaces those values with mean to reduce bias.
```

[58]:
```
#Then we one-hot encode for catgeroical features
#Had to rename resulting columns to keep same index
#Then updated that inthe df
df_encoded = pd.get_dummies(df, columns=['host_location'])
encoded_columns = df_encoded.columns
column_to_keep = 'host_location'
new_column_names = [column_to_keep if col.startswith(column_to_keep) else col for col in encoded_columns]
df_encoded.columns = new_column_names
```

[39]:
```
#Winsorize df to handle outliers
winsorized_prices = winsorize(df['price'], limits=[0.05, 0.05])
df['price'] = winsorized_prices

#Since no feature renaming is necessary and there are no NLP features, we can move forward.
```

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.

[59]:
```
#The features I will choose are location, number of bedrooms, number of␣
↪bathrooms, availability, and accommodation type.
#Selecting features and target
selected_features = ['host_location', 'bedrooms', 'bathrooms',␣
↪'host_is_superhost', 'reviews_per_month']
X = df_encoded[selected_features]
y = df['price']
```

[61]:
```
#Splitting data into training and testing data sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[66]:
```
#I will now be implementing all of the models at once for better visualization␣
↪purposes.

#Initializing all models
linear_reg = LinearRegression()
decision_tree_reg = DecisionTreeRegressor()
random_forest_reg = RandomForestRegressor()
gradient_boosting_reg = GradientBoostingRegressor()

#Training all models
linear_reg.fit(X_train, y_train)
decision_tree_reg.fit(X_train, y_train)
random_forest_reg.fit(X_train, y_train)
gradient_boosting_reg.fit(X_train, y_train)
```

[66]:
```
GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.1, loss='ls', max_depth=3,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=100,
                          n_iter_no_change=None, presort='deprecated',
                          random_state=None, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)
```

[63]:
```
#Evaluating all models
y_pred_linear = linear_reg.predict(X_test)
y_pred_decision_tree = decision_tree_reg.predict(X_test)
y_pred_random_forest = random_forest_reg.predict(X_test)
y_pred_gradient_boosting = gradient_boosting_reg.predict(X_test)
```

[64]:
```
#I chose the RMSE as an appropriate metric for my model. It is a perfect metric␣
 ↪that is appropriate for a regression model.
#It measures by taking the square root of the average squared difference␣
 ↪between actual and predicted values.
#The lower the RSME, the better it is for the models performance.
#This metric will help measure the accuracy of the regression model.
#I will now caluclate the RMSE for each model
rmse_linear = np.sqrt(mean_squared_error(y_test, y_pred_linear))
rmse_decision_tree = np.sqrt(mean_squared_error(y_test, y_pred_decision_tree))
rmse_random_forest = np.sqrt(mean_squared_error(y_test, y_pred_random_forest))
rmse_gradient_boosting = np.sqrt(mean_squared_error(y_test,␣
 ↪y_pred_gradient_boosting))
```

[65]:
```
#I can now compare the performance of the different models.
#The model that has the lowest RMSE would be chosen.
#The final model will finally be trained and applied to make predictions on new␣
 ↪data.
best_model = min(rmse_linear, rmse_decision_tree, rmse_random_forest,␣
 ↪rmse_gradient_boosting)

print("Root Mean Squared Error (RMSE) for Linear Regression:", rmse_linear)
print("Root Mean Squared Error (RMSE) for Decision Tree:", rmse_decision_tree)
print("Root Mean Squared Error (RMSE) for Random Forest:", rmse_random_forest)
print("Root Mean Squared Error (RMSE) for Gradient Boosting:",␣
 ↪rmse_gradient_boosting)

print("Best Model (lowest RMSE):", best_model)
```

```
Root Mean Squared Error (RMSE) for Linear Regression: 5744269107582.645
Root Mean Squared Error (RMSE) for Decision Tree: 95.50094603509025
Root Mean Squared Error (RMSE) for Random Forest: 87.44426507093115
Root Mean Squared Error (RMSE) for Gradient Boosting: 80.44849852817276
Best Model (lowest RMSE): 80.44849852817276
```

As we can see from the printed results of the RSME for the models, the best model is Gradient Bossting with the lowest RMSE of 80.4485. The RMSE value for the linear regression model is extremely high suggesting that this model is not performing well on the data. The RMSE value for the decision tree model is relatively lower than the linear regression model, but still relatively high. The RMSE value for the random forest model is lower than both linear regression and decision tree models, indicating a better performance. The gradient boosting model shows the lowest RMSE value among all the models which suggests that gradient boosting has performed the best.