**Data:** The dataset utilized in this assignment is named "Housing\_Iowa," which can be accessed on Kaggle through the following link: Housing\_Iowa Dataset (https://www.kaggle.com/c/house-prices-advanced-regression-techniques/overview). This dataset encompasses a total of 81 variables and includes 1,460 observations. Within the set of variables, there is a numerical target variable denoted as "SalePrice," an identification variable labeled as "ID," and seventy-nine predictor variables.

The target variable "SalePrice" exhibits skewness; nevertheless, this can be a significant issue when we try to build a multiple regression model. Furthermore, typical data issues, including missing values across several predictors, skewness in numerous numerical predictors, and the potential presence of outliers, need to concern us as well since data challenges are likely to significantly affect the performance of the fitted models when utilizing multiple regression algorithms. It worth to know that the missing value symbol is “NA” in this data set.

**Assignment Goal:** The objective of this concluding examination is to assess your comprehension of data preparation challenges in constructing a meaningful and effective regression model using LASSO through appropriate data preparation to fellow these steps.

**Step 1: Get Data (20 Points)**

Incorporating data into the software system utilized to construct this model can be achieved through various methods.

**Note: Put your programming coding in Appendix 1 to demonstrate how to obtain the data in the system.**

**Step 2: Data Exploration (50 Points)**

Data exploration is a critical step in the model-building process. For this assignment, we like you to produce the following tables.

1. Table 1: This table includes “Variable Name”, “Missing Count”, “Missing Percentage”, “Skewness”, “Mean”, “Median”, “Minimum”, and “Maximum” for all numerical predictors.

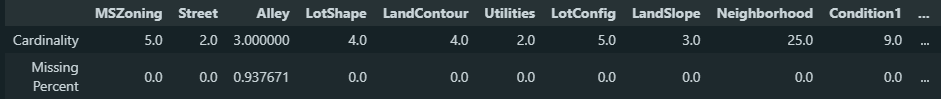
**(See ipynb for full table)**

A screenshot of a computer

Description automatically generated

1. Table 2: This table includes “Variable Name”, “Cardinality”, and “Missing Percentage” for all character predictors.

**(See ipynb for full table)**



Please incorporate a summary table for each component of STEP #2 within this response. Additionally, include the corresponding code in Appendix 2.

**Step 3: Data Preparation (100 Points)**

1. Create one missing value indicator for each predictor

(see full in ipynb)

A screenshot of a computer

Description automatically generated

1. Impute all numerical predictors with missing value using simple mean imputation

(see full in ipynb)

A screenshot of a computer

Description automatically generated

1. Perform adequate transformation for all skewed numerical predictors after imputation and produce a table that has the following columns: “Variable Name”, “Transformation Power”, “Skewness Before Transformation”, and “Skewness after Transformation”

A screenshot of a computer

Description automatically generated

1. Prepare all character predictors with cardinality higher than 6 using smoothing mean discussed in class

A screenshot of a graph

Description automatically generated

1. Skip outlier detection in this assignment

Please incorporate a summary table for each component of STEP #3 within this response. Additionally, include the corresponding code in Appendix 3.

**Step 4: Data Partition (30 Points)**

Considering the relatively modest size of this dataset, comprising 1,460 cases, we opt for a five-fold cross-validation approach for the purposes of this assignment. The complete dataset, denoted as B, is randomly partitioned into five distinct subsets:

**Please put your code for this part in Appendix 4.**

**Step 5: Five Multiple Regression Model Fitting (40 Points)**

Using the data partition outlined in Step 4, build five models by using four parts for training and reserving one part for testing. This process will allow for the computation of five distinct performance metrics, as detailed in Table 1.

|  |  |  |
| --- | --- | --- |
| **Table 1** | | |
| **Model** | **Training Data** | **Testing Data** |
| **I** |  |  |
| **II** |  |  |
| **II** |  |  |
| **IV** |  |  |
| **V** |  |  |

**Report the following information all five models built by completing Table 2 below.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Table 2** | | | |
| **Model** | **Number of Terms** | **Training MSE** | **Testing ASE** |
| **I** | **1168** | **279957210.2315116** | **630344624.6805023** |
| **II** | **1168** | **301960140.82225007** | **525104080.4214683** |
| **III** | **1168** | **323810991.77858937** | **454862015.5831138** |
| **IV** | **1168** | **326299264.9198287** | **384193151.79863673** |
| **V** | **1168** | **348787313.2843746** | **315723369.617615** |

**Please put your code for this part in Appendix 5.**

**Step 6: Ensemble all five models and calculate the MSE of this ensembled model (20 Points)**

The ultimate model is an ensemble formed by combining these five models, and the performance of this ensemble model can be estimated by calculate this ensembled model’s MSE.

462045448.4202672

**Step 7: Understanding the Results (40 Points)**

Write a report to report all your findings.

**Appendix 1**

# read in data

df = pd.read\_excel('House\_Iowa.xlsx')

**Appendix 2**

# creating numerical and categorical labels

labels = df.columns

df\_num = df.select\_dtypes(include=[np.number])

df\_num = df\_num.drop(['Id'], axis=1)

num\_labels = df\_num.columns

df\_cat = df.select\_dtypes(include=[object])

cat\_labels = df\_cat.columns

# creating lists for each column

missing\_count = []

missing\_percent = []

skewness = []

mean = []

median = []

minimum = []

maximum = []

cardinality = []

missing\_percent\_cat = []

# find desired table values

for label in num\_labels:

missing\_count.append(df[label].isnull().sum())

missing\_percent.append(df[label].isnull().sum()/len(df[label]))

skewness.append(df[label].skew())

mean.append(df[label].mean())

median.append(df[label].median())

minimum.append(df[label].min())

maximum.append(df[label].max())

for label in cat\_labels:

cardinality.append(len(df[label].unique()))

missing\_percent\_cat.append(df[label].isnull().sum()/len(df[label]))

# creating tables

table1 = pd.DataFrame({'Missing Count': missing\_count, 'Missing Percent': missing\_percent, 'Skewness': skewness, 'Mean': mean, 'Median': median, 'Minimum': minimum, 'Maximum': maximum}, index=num\_labels)

table2 = pd.DataFrame({'Cardinality': cardinality, 'Missing Percent': missing\_percent\_cat}, index=cat\_labels)

# swapping axes to desired

table1.swapaxes("index", "columns")

table2.swapaxes("index", "columns")

**Appendix 3**

# create missing value indicator columns

for label in num\_labels:

df[label + '\_missing'] = df[label].isnull().astype(int)

for label in cat\_labels:

df[label + '\_missing'] = df[label].isnull().astype(int)

# fill all num

imputer = SimpleImputer(strategy='mean')

df[num\_labels] = imputer.fit\_transform(df[num\_labels])

# selecting skewed columns

skew\_labels = []

pos\_skew\_labels = []

neg\_skew\_labels = []

for label in num\_labels:

if df[label].skew() > 0.5 or df[label].skew() < -0.5:

skew\_labels.append(label)

if df[label].skew() > 0.5:

pos\_skew\_labels.append(label)

else:

neg\_skew\_labels.append(label)

# transform skewed columns

for label in num\_labels:

if label in pos\_skew\_labels:

df[label + "\_transformed"] = np.log1p(df[label])

elif label in neg\_skew\_labels:

df[label + "\_transformed"] = np.power(df[label], 2)

# analyze skew transformation

fig = plt.figure(figsize=(20, 20))

for i in range(len(skew\_labels)):

ax = fig.add\_subplot(6, 6, i+1)

sns.histplot(df[skew\_labels[i]], ax=ax)

# produce before and after skewness tables

print("\nVariable Name", "\t Skewness Before", "\t Skewness After")

for label in pos\_skew\_labels:

print(label, "\t", df[label].skew(), "\t", df[label + "\_transformed"].skew())

for label in neg\_skew\_labels:

print(label, "\t", df[label].skew(), "\t", df[label + "\_transformed"].skew())

**Appendix 4**

# prepare data for modeling by dropping altered columns

if 'SalePrice' in skew\_labels:

skew\_labels.remove('SalePrice')

df.drop(skew\_labels, axis=1, inplace=True) # from \_transformed

df.drop(high\_card\_labels, axis=1, inplace=True) # from \_smoothed

df.dropna(axis=1, inplace=True) # from \_missing

# encoding remaining categorical columns

df = pd.get\_dummies(df, drop\_first=True, dtype=float)

# df to arrays

data = df.drop(['SalePrice'], axis=1).values

target = df['SalePrice'].values

# k-fold with 5 folds

kf = KFold(n\_splits=5, shuffle=True, random\_state=42)

# list subsets

for fold, (train\_index, test\_index) in enumerate(kf.split(data)):

X\_train, x\_test = data[train\_index], data[test\_index]

y\_train, y\_test = target[train\_index], target[test\_index]

print("Fold: ", fold + 1)

print("Train indices: ", train\_index)

print("Test indices: ", test\_index)

**Appendix 5**

# setting up model

model = Lasso(alpha=0.5)

ensemble = 0

i = 1

for train\_index, test\_index in kf.split(X):

X\_train, X\_test = data[train\_index], data[test\_index]

y\_train, y\_test = target[train\_index], target[test\_index]

model.fit(X\_train, y\_train)

y\_train\_pred = model.predict(X\_train)

y\_test\_pred = model.predict(X\_test)

print("Fold: ", i)

print("Train MSE:",mean\_squared\_error(y\_train, y\_train\_pred))

print("Test ASE:", mean\_squared\_error(y\_test, y\_test\_pred))

ensemble += mean\_squared\_error(y\_test, y\_test\_pred)

i += 1