

Assignment

Homework1

Arystan Tatishev, Maanas Peri

A CS5785 Homework Assignment



**CORNELL
TECH**

May 20, 2024

Coding Exercises

Part I: The Housing Prices

Problem 2

Give 3 examples of continuous and categorical features in the dataset; choose one feature of each type and plot the histogram to illustrate the distribution.

Solution.

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
```

```
1 training_data = pd.read_csv("house-prices-advanced-regression-techniques/
2 train.csv")
3 testing_data = pd.read_csv("house-prices-advanced-regression-techniques/
4 test.csv")
5 training_data.head()
```

```
1 continuous_features = training_data.select_dtypes(include=[np.number]) #
2 used chatgpt to figure out how to select
3 categorical_features = training_data.select_dtypes(exclude=[np.number])
4 continuous_features.head()
5 categorical_features.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoS
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	

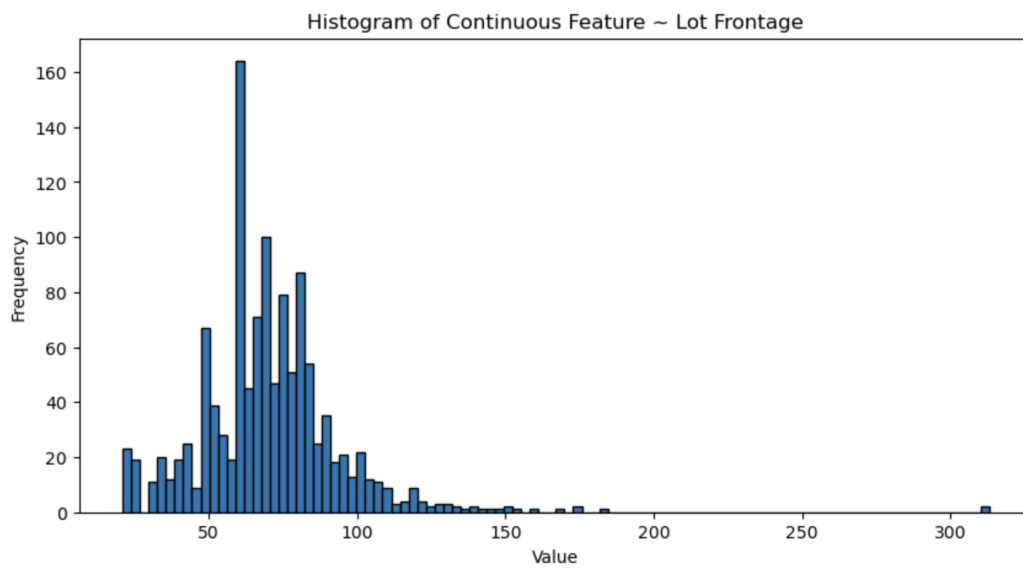
5 rows x 81 columns

	MSZoning	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	...	GarageType	GarageFinish	GarageQual	Gar
0	RL	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	...	Attchd	RFn	TA	
1	RL	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	Veenker	Feedr	...	Attchd	RFn	TA	
2	RL	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	...	Attchd	RFn	TA	
3	RL	Pave	NaN	IR1	Lvl	AllPub	Corner	Gtl	Crawfor	Norm	...	Detchd	Unf	TA	
4	RL	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NoRidge	Norm	...	Attchd	RFn	TA	

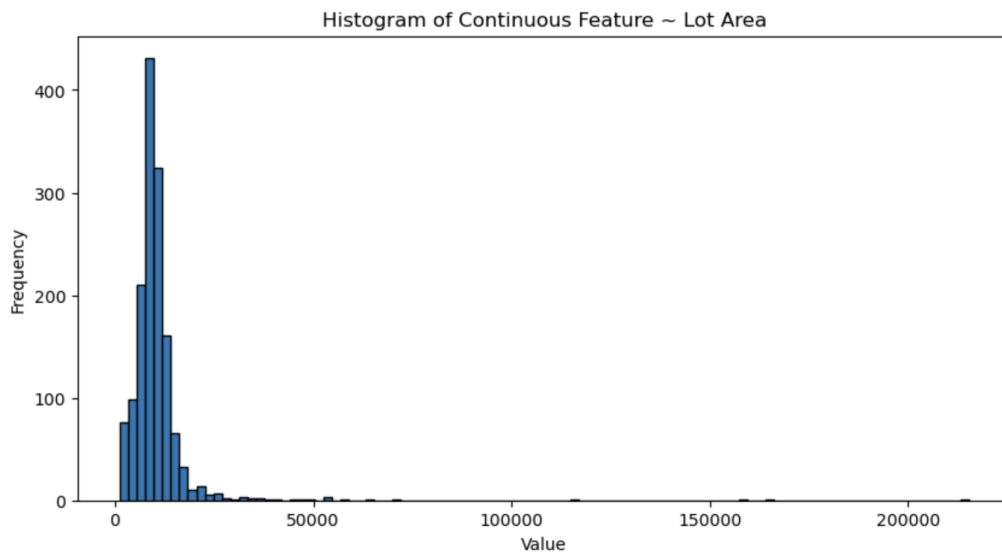
```
1 #selecting 3 variables - 2 continuous variables and 1 categorical variable
2 lotArea = "LotArea"
3 lotFrontage = "LotFrontage"
4 lotConfig = "LotConfig"
```

Here are the histograms of the 3 examples of continuous and categorical variables shown earlier.

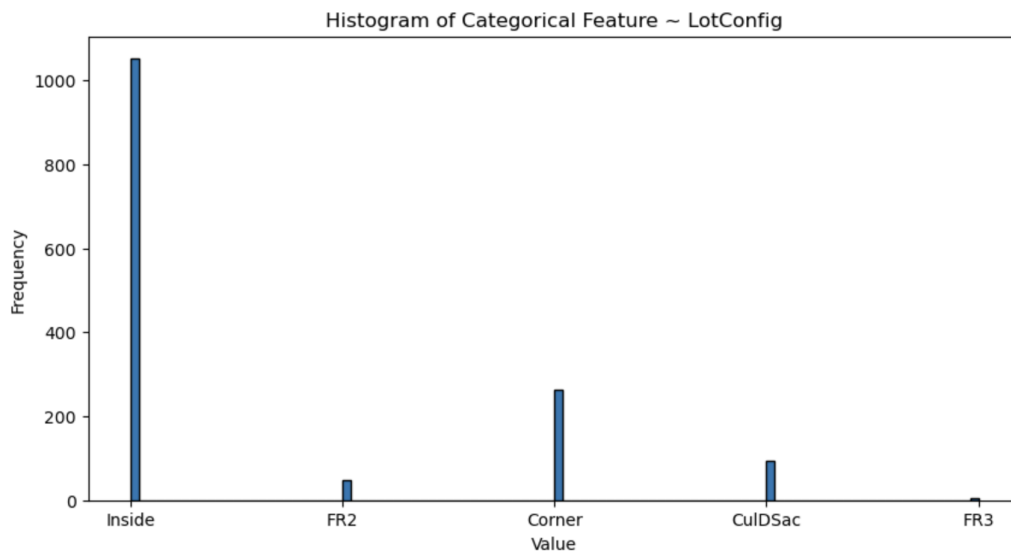
```
1 plt.figure(figsize=(10, 5))
2 plt.hist(training_data[lotFrontage], bins=100, edgecolor='k')
3 plt.title('Histogram of Continuous Feature ~ Lot Frontage')
4 plt.xlabel('Value')
5 plt.ylabel('Frequency')
6 plt.show()
```



```
1 plt.figure(figsize=(10, 5))
2 plt.hist(training_data[lotArea], bins=100, edgecolor='k')
3 plt.title('Histogram of Continuous Feature ~ Lot Area')
4 plt.xlabel('Value')
5 plt.ylabel('Frequency')
6 plt.show()
```



```
1 plt.figure(figsize=(10, 5))
2 plt.hist(training_data[lotConfig], bins=100, edgecolor='k')
3 plt.title('Histogram of Categorical Feature ~ LotConfig')
4 plt.xlabel('Value')
5 plt.ylabel('Frequency')
6 plt.show()
```



Problem 3

Pre-process your data, explain your pre-processing steps, and the reasons why you need them. (Hint: data pre-processing steps can include but are not restricted to: dealing with missing values, normalizing numerical values, dealing with categorical values etc.)

Solution.

First we are going to print out our training data information and see the count of the column variables that have non-null values. Our objective is to get the non-null count of all these columns up to a 1460 count.

- We need to first calculate if we even want to consider some columns, since they might be mostly made up of only a handful of non-null values. We can compute the percentage of missing values, and if consists of more than 33% of the values, then we can eliminate the column completely, since we won't be able to extrapolate/learn much from it
- Next, of the columns that we want to process (and only have a few null values) we need to handle missing continuous and categorical data separately. For missing continuous values, we need to replace them with the mean of the columns as to not affect the distribution of the continuous variable and serve as a consistent representative of the feature. Next, we are going to encode our categorical features into a numerical format (0s and 1s) to make it compatible with our model. This transformation will allow our model to work with our categorical data effectively.

Incorporating these steps ensures that our model can derive meaningful patterns and make accurate predictions.

```
1 print(training_data.info())
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
#   Column              Non-Null Count  Dtype
---  -
0   Id                   1460 non-null   int64
1   MSSubClass           1460 non-null   int64
2   MSZoning             1460 non-null   object
3   LotFrontage          1201 non-null   float64
4   LotArea              1460 non-null   int64
5   Street               1460 non-null   object
6   Alley                91 non-null     object
7   LotShape             1460 non-null   object
8   LandContour          1460 non-null   object
9   Utilities            1460 non-null   object
10  LotConfig            1460 non-null   object
11  LandSlope            1460 non-null   object
12  Neighborhood         1460 non-null   object
13  Condition1           1460 non-null   object
14  Condition2           1460 non-null   object
15  BldgType             1460 non-null   object
16  HouseStyle           1460 non-null   object
17  OverallQual          1460 non-null   int64
18  OverallCond          1460 non-null   int64
19  YearBuilt            1460 non-null   int64
20  YearRemodAdd         1460 non-null   int64
21  RoofStyle            1460 non-null   object
22  RoofMatl             1460 non-null   object
23  Exterior1st          1460 non-null   object
24  Exterior2nd          1460 non-null   object
25  MasVnrType           1452 non-null   object
26  MasVnrArea           1452 non-null   float64
27  ExterQual            1460 non-null   object
28  ExterCond            1460 non-null   object
29  Foundation           1460 non-null   object
30  BsmtQual             1423 non-null   object
31  BsmtCond             1423 non-null   object
32  BsmtFinType1         1423 non-null   object
33  BsmtFinType2         1422 non-null   object
34  BsmtFinSF1           1460 non-null   int64
35  BsmtFinType2         1422 non-null   object
36  BsmtFinSF2           1460 non-null   int64
37  BsmtUnfSF            1460 non-null   int64
38  TotalBsmtSF          1460 non-null   int64
39  Heating              1460 non-null   object
40  HeatingQC            1460 non-null   object
41  CentralAir           1460 non-null   object
42  Electrical           1459 non-null   object
43  1stFlrSF             1460 non-null   int64
44  2ndFlrSF             1460 non-null   int64
45  LowQualFinSF         1460 non-null   int64
46  GrLivArea            1460 non-null   int64
47  BsmtFullBath         1460 non-null   int64
48  BsmtHalfBath         1460 non-null   int64
49  FullBath             1460 non-null   int64
50  HalfBath             1460 non-null   int64
51  BedroomAbvGr         1460 non-null   int64
52  KitchenAbvGr         1460 non-null   int64
53  KitchenQual          1460 non-null   object
54  TotRmsAbvGrd         1460 non-null   int64
55  Functional           1460 non-null   object
56  Fireplaces           1460 non-null   int64
57  FireplaceQu          770 non-null    object
58  GarageType           1379 non-null   object
59  GarageYrBlt          1379 non-null   float64
60  GarageFinish         1379 non-null   object
61  GarageCars           1460 non-null   int64
62  GarageArea           1460 non-null   int64
63  GarageQual           1379 non-null   object
64  GarageCond           1379 non-null   object
65  PavedDrive           1460 non-null   object
66  WoodDeckSF           1460 non-null   int64
67  OpenPorchSF          1460 non-null   int64
68  EnclosedPorch        1460 non-null   int64
69  3SanPorch            1460 non-null   int64
70  ScreenPorch          1460 non-null   int64
71  PoolArea             1460 non-null   int64
72  PoolQC               7 non-null     object
73  Fence                281 non-null   object
74  MiscFeature          54 non-null    object
75  MiscVal              1460 non-null   int64
76  MoSold               1460 non-null   int64
77  YrSold               1460 non-null   int64
78  SaleType             1460 non-null   object
79  SaleCondition         1460 non-null   object
80  SalePrice            1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
None

```

```

1 processing_data = training_data
2 missing_percentage = (processing_data.isnull().sum() / len(training_data)) *
    100
3 columns_to_remove = missing_percentage[missing_percentage > 33].index.tolist()
4 # print(columns_to_remove)
5 cleaned_data = processing_data.drop(columns=columns_to_remove)
6 # print(cleaned_data.info())
7
8 #replace the continuous variables from null values to mean()
9 continuous_features = cleaned_data.select_dtypes(include=[np.number])
10 categorical_features = cleaned_data.select_dtypes(exclude=[np.number])
11 #replace the discrete variables from null values to
12
13 #processing the continuous features first, by filling in all the null values
    with the mean of values, this way the distribution of this data won't get
    affected
14 mean_values = continuous_features.mean()
15 continuous_features.fillna(mean_values, inplace=True)
16 # print(continuous_features.info()) #now we filled in the missing naan values
    with the mean value

```

Problem 4

One common method of pre-processing categorical features is to use a one-hot encoding (OHE).

Suppose that we start with a categorical feature x_j , taking three possible values:

$x_j \in \{R, G, B\}$. A one-hot encoding of this feature replaces x_j with three new features: x_{jR}, x_{jG}, x_{jB} . Each feature contains a binary value of 0 or 1, depending on the value taken by x_j . For example, if $x_j = G$, then $x_{jG} = 1$ and $x_{jR} = x_{jB} = 0$.

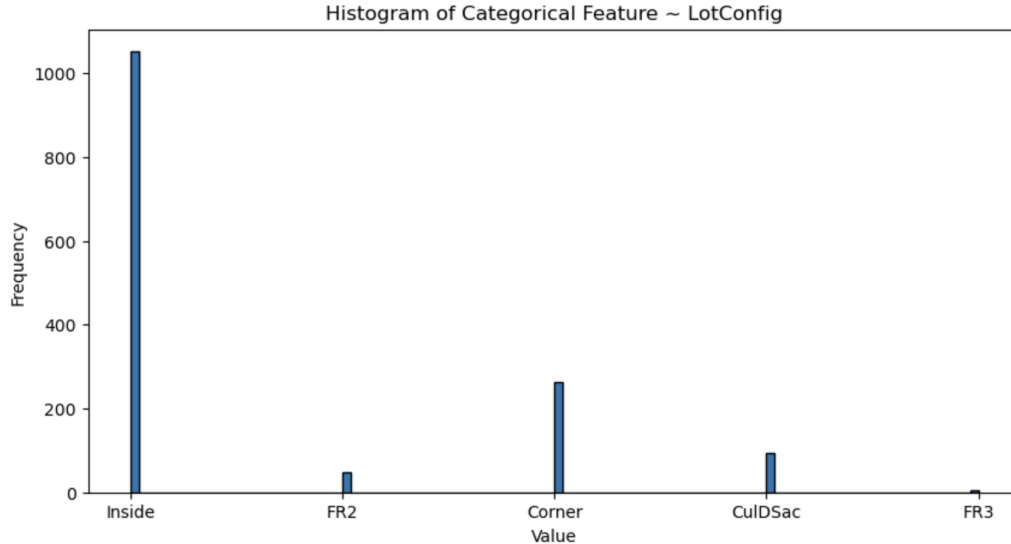
Give some examples of features that you think should use a one-hot encoding and explain why. Convert at least one feature to a one-hot encoding (you can use your own implementation, or that in pandas or scikit-learn) and visualize the results by plotting feature histograms of the original feature and its new one-hot encoding.

The following list of categorical features that need encoding are below:

```
1 ['MSZoning', 'Street', 'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual', 'Functional', 'GarageType', 'GarageFinish', 'GarageQual', 'GarageCond', 'PavedDrive', 'SaleType', 'SaleCondition']
```

We chose one-hot encoding for all of these features because all of them have discrete values that cannot have any arithmetic operations performed on them. If 'LotConfig' = 'Inside', our model is going to have a hard time performing variant and loss operations on it (essentially having issues quantifying this 'Reg value'). That's why it's important to create new subcategories and use binary values to determine the proper 'LotConfig' attributes.

For example, the original representation of the LotConfig is as we discussed above:



After performing a one-hot encoding program (below) on this specific categorical feature, we can understand how this histogram transformed, thereby reflecting the data.

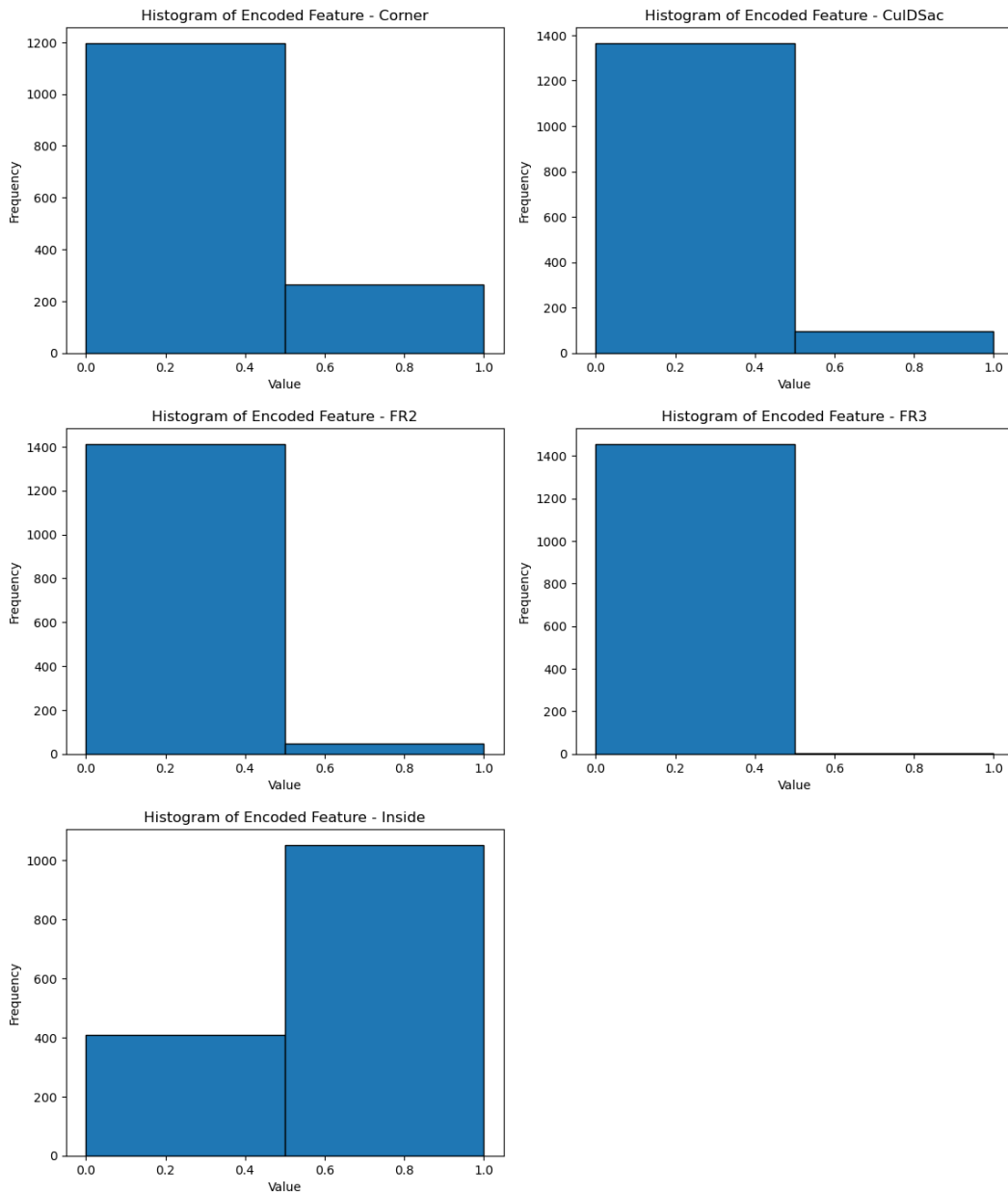
```
1 newCategoricalData = pd.DataFrame()
2 newCategoricalData = pd.get_dummies(training_data[lotConfig])
3
4 for column in newCategoricalData.columns:
5     plt.hist(newCategoricalData[column], bins=2, edgecolor='k') # Assuming
6     binary encoded (0s and 1s)
```

```

6 plt.title(f'Histogram of Encoded Feature - {column}')
7 plt.xlabel('Value')
8 plt.ylabel('Frequency')
9 plt.show()

```

This gives us the following graphs:



Now, we have 5 different subclasses that have transformed their data to reflect their categorical data numerically.

The code for one-hot encoding is presented below

Solution.

```

1 #performing encoding on all categorical features, and this is the only
  processing we will need

```

```

2 encoded_columns = pd.DataFrame()
3 encoded_columns = pd.get_dummies(categorical_features)
4 # print(encoded_columns.head())
5
6 #concatenating the processed categorical and continuous data into 1 big pandas
  dataframe
7 finalData = pd.concat([encoded_columns, continuous_features], axis=1)
8 # print(finalData.info())

```

Problem 5

Using ordinary least squares (OLS), try to predict house prices on this dataset. Choose the features (or combinations of features) you would like to use or ignore, provided you justify your choice. Evaluate your predictions on the training set using the MSE and the R^2 score. For this question, you need to implement OLS from scratch without using any external libraries or packages.

Solution.

The features that we chose to drop are the initial columns that surpass the 33% threshold of missing values. These are: ['Alley', 'FireplaceQu', 'PoolQC', 'Fence', 'MiscFeature']. Eventually, we also drop the 'Id' feature as well as the 'SalePrice' to prevent target leakage (and instead train it as a Y-Value in our model as displayed in the program below). This means that we are left with the remaining features shown below that would help us evaluate our training data sets.

```

1 ['MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street', 'LotShape', '
  LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood', '
  Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'OverallQual', '
  OverallCond', 'YearBuilt', 'YearRemodAdd', 'RoofStyle', 'RoofMatl', '
  Exterior1st', 'Exterior2nd', 'MasVnrType', 'MasVnrArea', 'ExterQual', '
  ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', '
  BsmtFinType1', 'BsmtFinSF1', 'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', '
  TotalBsmtSF', 'Heating', 'HeatingQC', 'CentralAir', 'Electrical', '1
  stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', '
  BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', '
  KitchenQual', 'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'GarageType', '
  GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual', '
  GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '
  3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', '
  SaleType', 'SaleCondition']

```

After running this code for our training data:

```

1 import numpy as np
2 # Our parameters that minimize the mean squared objective for all of our
  columns
3 theta_best = np.linalg.solve(finalData.T.dot(finalData), finalData.T.dot(
  Y_train))
4 # print(theta_best)

```

After processing our testing data (described in the below question)


```

1 from sklearn.metrics import r2_score
2 y_train_pred = finalData.dot(theta_best)
3 print(r2_score(Y_train, y_train_pred))

```

resulted in a 0.9310337512792708 R-Score value, and a MSE score of 434955550.2062558, which makes some sense since we are dealing with houses that cost hundreds of thousands of dollars, and the difference is squared. These values are shown in our Jupyter Notebook.

Problem 6

Train your model using all of the training data (all data points, but not necessarily all the features), and test it using the testing data. Submit your results to Kaggle.

Solution.

We processed our testing data the same way we processed our training data, and there are many overlaps in the feature engineering choices, so for the purposes of conciseness, we won't repeat the same explanations again. Here is what our pre-processing steps for our test data looks like:

```

1 # Read test data
2 X_test = pd.read_csv('house-prices-advanced-regression-techniques/test.csv')
3
4 # drop
5 processing_testing_data = X_test
6 missing_percentage_test = (processing_testing_data.isnull().sum() / len(X_test
7                             )) * 100
8 columns_to_remove_test = missing_percentage_test[missing_percentage_test >
9                                     33].index.tolist()
10 # print(columns_to_remove)
11 cleaned_data_test = processing_testing_data.drop(columns=
12                                     columns_to_remove_test)
13
14 # fill na with mean values
15 mean_values_test = continuous_features_test.mean()
16 continuous_features_test.fillna(mean_values_test, inplace=True)
17
18 # one-hot encoding
19 encoded_columns_test = pd.DataFrame()
20 encoded_columns_test = pd.get_dummies(categorical_features_test)
21
22 # remove Id
23 finalData_test = pd.concat([encoded_columns_test, continuous_features_test],
24                             axis=1)
25 for col in finalData.columns:
26     if col not in finalData_test.columns:
27         finalData_test[col] = 0
28     print(col)

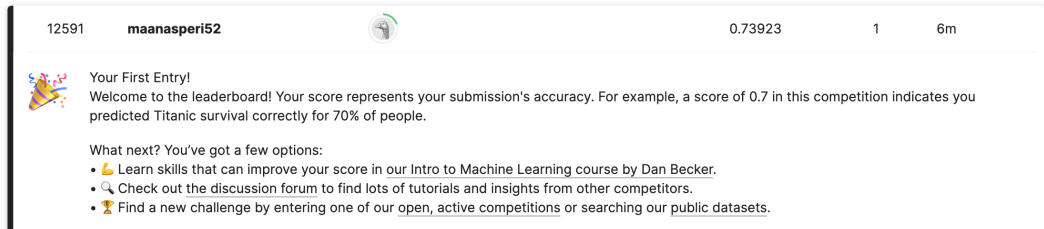
```

```


28
29 finalData_test.drop('Id',axis=1,inplace = True)
30 finalData_test['one'] = 1
31 #Reordering our test columns with the training data set to avoid incompatible
   columns error
32 finalData_test = finalData_test[finalData.columns]
33
34 print(finalData_test.shape)
35 finalData_test.head()

```

We submitted our solution to the Kaggle Competition, here's our results:



12591 **maanasperi52** 0.73923 1 6m

 Your First Entry!
Welcome to the leaderboard! Your score represents your submission's accuracy. For example, a score of 0.7 in this competition indicates you predicted Titanic survival correctly for 70% of people.

What next? You've got a few options:

- 🔥 Learn skills that can improve your score in our [Intro to Machine Learning course by Dan Becker](#).
- 🔍 Check out the [discussion forum](#) to find lots of tutorials and insights from other competitors.
- 🏆 Find a new challenge by entering one of our [open, active competitions](#) or searching our [public datasets](#).

Part II: The Titanic Disaster

Problem 2

Implement logistic regression (it's ok to use sklearn or similar software packages), try to predict whether a passenger survived the disaster with your model. Choose the features (or combinations of features) you would like to use or ignore, provided you justify your choice.

Solution. The features we chose to ignore were:

- Name (technically, we included it just didn't encode this discrete variable)
- Cabin
- Ticket

The reasoning for dropping these features was because they do not really provide any useful information on the probability of survival of the passenger

Problem 3

Train your classifier using all of the training data, and test it using the testing data. Submit your results to Kaggle.

Solution.

This is after our processing data, which is explained in detail in our jupyter notebook. We used the RandomForestClassifier because we wanted our model to replicate the unpredictable


nature of a dense forest/data, where each tree in the forest represents a unique decision-making process. This may lead to overfitting, however, and in the future, we aim to allocate a bigger portion of the dataset for evaluation methods.

```

1 from sklearn.ensemble import RandomForestClassifier
2
3 #Now we can finally train and fit the model to our dataset
4 x_train = training.iloc[:,1:]
5 y_train = training.iloc[:,0:1]
6 x_test = testing.iloc[:,:]
7
8 model = RandomForestClassifier(n_estimators = 70)
9
10 x_train.columns = x_train.columns.astype(str)
11 y_train.columns = y_train.columns.astype(str)
12 x_test.columns = x_test.columns.astype(str)
13
14 model.fit(x_train,y_train)
15 y_pred= model.predict(x_test)
16 accuracy = model.score(x_train, y_train)
17
18 print(accuracy)

```

0.9820022497187851 was our scoring accuracy, as in 98.2%, and after submitting to the Kaggle Competition, this was our leadership rankings.

12645	maanasperi52	0.73923	1	1d
 <p>Your First Entry! Welcome to the leaderboard! Your score represents your submission's accuracy. For example, a score of 0.7 in this competition indicates you predicted Titanic survival correctly for 70% of people.</p> <p>What next? You've got a few options:</p> <ul style="list-style-type: none"> 🎓 Learn skills that can improve your score in our Intro to Machine Learning course by Dan Becker. 🔍 Check out the discussion forum to find lots of tutorials and insights from other competitors. 🏆 Find a new challenge by entering one of our open, active competitions or searching our public datasets. 				

Written Exercises

1 Conceptual questions

Problem: (a)

Identify one advantage and one disadvantage of using gradient descent as the optimizer of a supervised linear model, compared to using the analytical formula based on the normal equations (as is done in the Ordinary Least Squares algorithm)

Solution.

Advantage:

Its great from an educational point of view because it teaches the concept of machine learning at a mathematical level that is understandable, and a concept that translates well to neural networks.

Its excellent for large datasets, because you can partition data into chunks, and it can manage the computational demands of large data more effectively than other optimizing algorithms. This does not mean that its fast though, as discussed below in the disadvantages section.

Disadvantage:

Iteratively approaching the minimum takes a lot of time. Even with alternative methods of gradient descent and parallel computing, its a relatively slow algorithm to keep getting closer and closer to the desired minimum loss.

Gradient descent still does not arrive at the exact minimum value, even through multiple iterations. This is because arriving at the exact minimum is only as precise as the learning rate alpha.

Problem: (b)

Imagine doing a regression problem. You discretize the output space of y into a large number of small intervals and apply a multi-class classification algorithm (like softmax regression) to predict the interval containing the target output. Would this approach be sufficient to solve the regression problem? Describe why or why not.

Solution.

Discretization is also known as the process of binning (dividing the range of the continuous variables into intervals), and this is a common method to train data. As long as the the thresholds of identifying the interval boundaries are accurate, this approach would be sufficient to solve the regression problem. Otherwise, poorly chosen intervals can make this algorithm make poor predictive choices when classifying data.

Using a multi-class classification algorithm, like softmax regression, is also optimal because regression algorithms are able to train faster and better using discrete values. Without these discrete values, the algorithms would have to exhaustively go through all the features for one value, whereas binning features would expedite the process.

Problem: (c)

Name one advantage and one disadvantage of performing multi-class classification by training multiple one-vs-all classifiers compared to directly using a multi-class algorithm like softmax regression.

Solution.**Advantage:**

Training a one-vs-all algorithm prepares a binary classifier for every single K classification type. These classifiers are trained separately for each different type of outcome, and the collection of these classifiers results in a multi-classification solution. This makes interpreting the values simple because of the simplified nature of binary classifiers, and easy to determine

which classifier is providing more inaccurate results than the other. Furthermore, each of these classifiers is compatible with a number of models, such as linear models, decision trees, logistic regression, etc. Last but not least, each binary classifier can have its own parameters, allowing for more control over each and every classifier.

Disadvantage:

The biggest disadvantage of a one-vs-all classifier is the class imbalance that occurs. For example, if there is a 90-10% distribution of Dataset A to Dataset B, then there won't be enough conclusive data to understand the prediction patterns of Dataset B. Classifiers are prone to struggle against skewed data. Even in the case of a 5-class classification problem, if there are 5 classes with a roughly equal 20% dataset distribution, each of these 5 classes ends up with a 20-80% distribution (class vs rest of classes).

Problem: (d)

The computational complexity of polynomial regression depends on the number of data points in the training set and on the degree of the polynomial that we are trying to fit.

- (i.) Assuming that each data point has d attributes and we want our model class to consist of all polynomials of d variables and degree at most p , state the dimension of the polynomial features that are needed to implement this model class. Specifying a big-Oh estimate is enough.
- (ii.) State the computational complexity of applying polynomial least squares with the above set of polynomial features. Again, a big-Oh estimate is sufficient.
- (iii.) Comment on how the above findings may influence your decision for when to apply polynomial regression in real-world settings.

Solution.

- (i.) $O(p^d)$
- (ii.) $O(n * p^d)$
- (iii.) It is prohibitively expensive to use high-degree polynomial regression given a large dataset with large number of attributes, there needs to be a trade-off

2 Analytical solution for Ordinary Least Squares

Problem

Analytical solution for Ordinary Least Squares. Consider a simple dataset of n training instances $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})\}$, with inputs $x^{(i)} \in \mathbb{R}$ and targets $y^{(i)} \in \mathbb{R}$. We are going to fit a simple linear model with parameters $\theta = (\theta_0, \theta_1)$ on this dataset:

$$f_{\theta}(x) = \theta_0 + \theta_1 x^{(i)}.$$

We define the learning objective to be the residual sum of squares, parameterized by θ_0, θ_1 :

$$J(\theta_0, \theta_1) = \sum_{i=1}^n (y^{(i)} - f_{\theta}(x)^{(i)})^2$$

Instead of using gradient descent, which works in an iterative manner, we will derive a formula for the parameters that minimize the objective J .

Problem: (a)

Calculate the partial derivatives $\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$ and $\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$.

Solution.

$$\begin{aligned} \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) &= \frac{\partial}{\partial \theta_0} \sum_{i=1}^n (y^{(i)} - f_{\theta}(x)^{(i)})^2 \\ &= \frac{\partial}{\partial \theta_0} \sum_{i=1}^n (y^{(i)} - \theta_0 - \theta_1 x^{(i)})^2 \\ &= \sum_{i=1}^n -2 * (y^{(i)} - \theta_0 - \theta_1 x^{(i)}) \\ &= \boxed{2 \sum_{i=1}^n (\theta_0 + \theta_1 x^{(i)} - y^{(i)})} \end{aligned} \tag{1}$$

$$\begin{aligned} \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) &= \frac{\partial}{\partial \theta_1} \sum_{i=1}^n (y^{(i)} - f_{\theta}(x)^{(i)})^2 \\ &= \frac{\partial}{\partial \theta_1} \sum_{i=1}^n (y^{(i)} - \theta_0 - \theta_1 x^{(i)})^2 \\ &= \sum_{i=1}^n -2x^{(i)} * (y^{(i)} - \theta_0 - \theta_1 x^{(i)}) \\ &= \boxed{2 \sum_{i=1}^n x^{(i)} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})} \end{aligned} \tag{2}$$

Problem: (b)

Consider the fact that $J(\theta_0, \theta_1)$ has an unique optimum ¹, which we denote as θ_0^*, θ_1^* . We can obtain the analytical solution for θ_0^*, θ_1^* by setting the gradient of J to zero, which yields the following normal equations:

$$\frac{\partial}{\partial \theta_0} J(\theta_0^*, \theta_1) = 0$$

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1^*) = 0$$

Write out the above equations and use them to prove the following properties:

$$\theta_0^* = \bar{y} - \theta_1 \bar{x}$$

and

$$\theta_1^* = \frac{\sum_{i=1}^n x^{(i)} (y^{(i)} - \bar{y})}{\sum_{i=1}^n x^{(i)} (x^{(i)} - \bar{x})}$$

(Note: $\bar{x} = \frac{1}{n} \sum_{i=1}^n x^{(i)}$ and $\bar{y} = \frac{1}{n} \sum_{i=1}^n y^{(i)}$.)

¹ $J(\theta_0, \theta_1)$, the cost function for linear regression, has a unique optimum (i.e., a global minimum). The technical reason for this fact is that J is a convex function, which can be verified, for example, by showing that its Hessian is positive semidefinite.

Solution.

$$\frac{\partial}{\partial \theta_0} J(\theta_0^*, \theta_1) = 2 * \sum_{i=1}^n (\theta_0^* + \theta_1 x^{(i)} - y^{(i)}) = 0$$

$$\sum_{i=1}^n \theta_0^* + \sum_{i=1}^n \theta_1 x^{(i)} - \sum_{i=1}^n y^{(i)} = 0$$

$$\sum_{i=1}^n \theta_0^* = \sum_{i=1}^n y^{(i)} - \sum_{i=1}^n \theta_1 x^{(i)}$$

$$\theta_0^* * n = \sum_{i=1}^n y^{(i)} - \sum_{i=1}^n \theta_1 x^{(i)}$$

$$\theta_0^* = \frac{1}{n} \sum_{i=1}^n y^{(i)} - \frac{1}{n} \sum_{i=1}^n \theta_1 x^{(i)}$$

$$\boxed{\theta_0^* = \bar{y} - \theta_1 \bar{x}} \tag{3}$$

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1^*) = 2 * \sum_{i=1}^n x^{(i)} * (\theta_0 + \theta_1^* x^{(i)} - y^{(i)}) = 0$$

$$\sum_{i=1}^n x^{(i)} * (\bar{y} - \theta_1 \bar{x} + \theta_1^* x^{(i)} - y^{(i)}) = 0$$

$$\sum_{i=1}^n \bar{y} x^{(i)} - \theta_1 \bar{x} x^{(i)} + \theta_1^* x^{2(i)} - y^{(i)} x^{(i)} = 0$$

$$\sum_{i=1}^n \theta_1^* x^{2(i)} - \theta_1 \bar{x} x^{(i)} = \sum_{i=1}^n y^{(i)} x^{(i)} - \bar{y} x^{(i)}$$

$$\theta_1 \sum_{i=1}^n x^{(i)} (x^{(i)} - \bar{x}) = \sum_{i=1}^n x^{(i)} (y^{(i)} - \bar{y})$$

$$\theta_1^* = \frac{\sum_{i=1}^n x^{(i)} (y^{(i)} - \bar{y})}{\sum_{i=1}^n x^{(i)} (x^{(i)} - \bar{x})} \quad (4)$$

Problem: (c)

For the optimal θ_0^*, θ_1^* , calculate the sum of the residuals

$$\sum_{i=1}^n e^{(i)} = \sum_{i=1}^n (y^{(i)} - (\theta_0^* + \theta_1^* x^{(i)}))$$

. What can you learn from the value of $\sum_{i=1}^n e^{(i)}$?

Solution.

$$\begin{aligned}
 \sum_{i=1}^n e^{(i)} &= \sum_{i=1}^n (y^{(i)} - (\theta_0^* + \theta_1^* x^{(i)})) \\
 &= \sum_{i=1}^n (y^{(i)} - \bar{y} + \theta_1^* \bar{x} - \theta_1^* x^{(i)}) \\
 &= \sum_{i=1}^n (y^{(i)} - \bar{y} + \theta_1^* (\bar{x} - x^{(i)})) \\
 &= \sum_{i=1}^n \left(y^{(i)} - \bar{y} + \frac{x^{(i)} (y^{(i)} - \bar{y})}{x^{(i)} (x^{(i)} - \bar{x})} (\bar{x} - x^{(i)}) \right) \\
 &= \sum_{i=1}^n \left(y^{(i)} - \bar{y} + \cancel{\frac{x^{(i)}}{x^{(i)}}} \frac{(y^{(i)} - \bar{y})}{\cancel{(x^{(i)} - \bar{x})}} (\bar{x} - \cancel{x^{(i)}}) \right) \\
 &= \sum_{i=1}^n (2y^{(i)} - 2\bar{y}) \\
 &= 2 \sum_{i=1}^n (y^{(i)} - \bar{y}) \\
 &= 2n\bar{y} - 2\bar{y}n \\
 &= \boxed{0}
 \end{aligned} \quad (5)$$

Sum of residuals for linear regression model is zero, which means that the line runs perfectly in the middle of data points. So positive residuals cancel out negative residuals, thus the sum is 0.

3 Maximum Likelihood Estimation

Problem: (a)

You are conducting an experiment involving the tossing of a four-sided dice. You

conducted eight dice throws and meticulously recorded the outcomes. Your objective is to determine from this dataset the true probabilities of the dice falling on each of its four sides (which we denote by 1, 2, 3, 4).

The dataset $D = \{3, 2, 3, 4, 4, 4, 2, 4\}$, shown in Table 1, consists of the recorded outcomes of the eight dice throws. Each $x^{(i)} \in D$ refers to an individual throw. Each throw in Table 1 is represented by a throw number and a corresponding outcome (e.g., the dice landed on value '3' in throw number 1).

Throw Number	Outcome
1	3
2	2
3	3
4	4
5	4
6	4
7	2
8	4

Table 1: Results of Eight Sample Throws of the 4-sided Dice

Problem: i. Probabilistic Model.

Construct a probabilistic model that captures the underlying probabilities governing the dice outcomes. Define the model and its parameters, aiming to establish the probabilities associated with each outcome.

Solution.

- There are four possible outcomes: 1, 2, 3, 4. A training dataset consists dice throws, $D = \{3, 2, 3, 4, 4, 4, 2, 4\}$
- Our task is to estimate the true probability of each dice throw
- That being said, it means that we will have 4 parameters
 - $P_{\theta}(x = j) : \theta_j$ where $\theta_j \in [0, 1]$

Problem: ii. Learning Paradigm.

Classify the experiment as fitting into supervised learning, unsupervised learning, or reinforcement learning paradigms.

Solution. The experiment fits into supervised learning paradigm

Problem: (b)

Write a formula for the log-likelihood of the dataset under the probabilistic model. Provide an intuitive argument for why we would want to optimize this objective.

Solution.

$$\begin{aligned}
 L(\theta) &= \prod_{i=1}^n P_{\theta}(x^{(i)}) \\
 &= \theta_3 \cdot \theta_2 \cdot \theta_3 \cdot \theta_4 \cdot \theta_4 \cdot \theta_4 \cdot \theta_2 \cdot \theta_4 \\
 &= \theta^{\# \text{ 1's }} \cdot \theta^{\# \text{ 2's }} \cdot \theta^{\# \text{ 3's }} \cdot \theta^{\# \text{ 4's }} \\
 \log L(\theta) &= \log(\theta_1^{\# \text{ 1's }} \cdot \theta_2^{\# \text{ 2's }} \cdot \theta_3^{\# \text{ 3's }} \cdot \theta_4^{\# \text{ 4's }}) \\
 &= \# \text{ 1's } \cdot \log(\theta_1) + \# \text{ 2's } \cdot \log(\theta_2) + \# \text{ 3's } \cdot \log(\theta_3) + \# \text{ 4's } \cdot \log(\theta_4) \\
 &= 2 \cdot \log(\theta_2) + 2 \cdot \log(\theta_3) + 4 \cdot \log(\theta_4)
 \end{aligned} \tag{6}$$

We want to optimize the probabilities of the observed dataset.

Problem: (c)

Calculate the maximum likelihood estimate of the model parameters. Interpret the parameter values within the context of dice toss probabilities, considering how they relate to the observed outcomes.

Solution.

$$\begin{aligned}
 \frac{\partial \log L(\theta)}{\partial \theta_1} &= 0 \\
 \frac{\partial \log L(\theta)}{\partial \theta_2} &= \frac{2}{\theta_2} \\
 \frac{\partial \log L(\theta)}{\partial \theta_3} &= \frac{2}{\theta_3} \\
 \frac{\partial \log L(\theta)}{\partial \theta_4} &= \frac{4}{\theta_4}
 \end{aligned} \tag{7}$$

$$\frac{2}{\theta_2} = \frac{2}{\theta_3} = \frac{4}{\theta_4}, \text{ where } \sum_{i=1}^4 = 1$$

$$\theta_1 = 0, \theta_2 = 0.25, \theta_3 = 0.25, \theta_4 = 0.5$$

Problem: (d.)

Recognize scenarios where this approach might yield inaccurate estimates of the true probabilities of the dice falling on each of its four sides. Provide at least one example and elucidate the reasoning behind potential inaccuracies.

Solution. Small data sets. Theoretically it makes sense if a 4 sided die will have an equal probability to land on each of its sides. However, given the dataset, this approach suggests other probabilities, akin to a loaded die. Increasing the dataset will remove these inaccuracies.