Introduction
Thousands of houses are sold everyday. There are some questions every buyer asks himself like: What is the actual price that this house deserves? Am I paying a fair price? In this paper, a machine learning model is proposed to predict a house price based on data related to the house (its size, the year it was built in, etc.). During the development and evaluation of our model, we will show the code used for each step followed by its output. This will facilitate the reproducibility of our work. In this study, Python programming language with a number of Python packages will be used.

Goals of the Study
The main objectives of this study are as follows:

To apply data preprocessing and preparation techniques in order to obtain clean data
To build machine learning models able to predict house price based on house features
To analyze and compare models performance in order to choose the best model

Paper Organization
This paper is organized as follows: in the next section, section 2, we examine studies related to our work from scientific journals. In section 3, we go through data preparation including data cleaning, outlier removal, and feature engineering. Next in section 4, we discuss the type of our problem and the type of machine-learning prediction that should be applied; we also list the prediction techniques that will be used. In section 5, we choose algorithms to implement the techniques in section 4; we build models based on these algorithms; we also train and test each model. In section 6, we analyze and compare the results we got from section 5 and conclude the paper.

Literature Review
In this section, we look at five recent studies that are related to our topic and see how models were built and what results were achieved in these studies.

Stock Market Prediction Using Bayesian-Regularized Neural Networks
In a study done by Ticknor (2013), he used Bayesian regularized artificial neural network to predict the future operation of financial market. Specifically, he built a model to predict future stock prices. The input of the model is previous stock statistics in addition to some financial technical data. The output of the model is the next-day closing price of the corresponding stocks.

The model proposed in the study is built using Bayesian regularized neural network. The weights of this type of networks are given a probabilistic nature. This allows the network to penalize very complex models (with many hidden layers) in an automatic manner. This in turn will reduce the overfitting of the model.

The model consists of a feedforward neural network which has three layers: an input layer, one hidden layer, and an output layer. The author chose the number of neurons in the hidden layer based on experimental methods.The input data of the model is normalized to be between -1 and

1, and this opertion is reversed for the output so the predicted price appears in the appropriate scale.

The data that was used in this study was obtained from Goldman Sachs Group (GS), Inc. and Microsoft Corp. (MSFT) . The data covers 734 trading days (4 January 2010 to 31 December 2012). Each instance of the data consisted of daily statistics: low price, high price, opening price, close price, and trading volume. To facilitate the training and testing of the model, this data was split into training data and test data with 80% and 20% of the original data, respectively. In addition to the daily-statistics variables in the data, six more variables were created to reflect financial indicators.

The performance of the model were evaluated using mean absolute percentage error (MAPE) performance metric. MAPE was calculated using this formula:

MAPE=∑ri=1(abs(yi−pi)/yi)r×100
where pi
 is the predicted stock price on day i
, yi
 is the actual stock price on day i
, and r
 is the number of trading days.

When applied on the test data, The model achieved a MAPE score of 1.0561 for MSFT part, and 1.3291 for GS part. Figure 1 shows the actual values and predicted values for both GS and MSFT data.

Figure 1: Predicted vs. actual price

Stock Market Prediction Using A Machine Learning Model

In another study done by Hegazy, Soliman, and Salam (2014), a system was proposed to predict daily stock market prices. The system combines particle swarm optimization (PSO) and least square support vector machine (LS-SVM), where PSO was used to optimize LV-SVM.

The authors claim that in most cases, artificial neural networks (ANNs) are subject to the overfitting problem. They state that support vector machines algorithm (SVM) was developed as an alternative that doesn't suffer from overfitting. They attribute this advantage to SVMs being based on the solid foundations of VC-theory. They further elaborate that LS-SVM method was reformulation of traditional SVM method that uses a regularized least squares function with equality constraints to obtain a linear system that satisfies Karush-Kuhn-Tucker conditions for getting an optimal solution.

The authors describe PSO as a popular evolutionary optimization method that was inspired by organism social behavior like bird flocking. They used it to find the optimal parameters for LS-SVM. These parameters are the cost penalty C

, kernel parameter γ
, and insensitive loss function ϵ
.

The model proposed in the study was based on the analysis of historical data and technical financial indicators and using LS-SVM optimized by PSO to predict future daily stock prices. The model input was six vectors representing the historical data and the technical financial indicators. The model output was the future price. The model used is represented in Figure 2.

Figure 2: The structure of the model used Networks
In a study done by Ticknor (2013), he used Bayesian regularized artificial neural network to predict the future operation of financial market. Specifically, he built a model to predict future stock prices. The input of the model is previous stock statistics in addition to some financial technical data. The output of the model is the next-day closing price of the corresponding stocks.

The model proposed in the study is built using Bayesian regularized neural network. The weights of this type of networks are given a probabilistic nature. This allows the network to penalize very complex models (with many hidden layers) in an automatic manner. This in turn will reduce the overfitting of the model.

The model consists of a feedforward neural network which has three layers: an input layer, one hidden layer, and an output layer. The author chose the number of neurons in the hidden layer based on experimental methods.The input data of the model is normalized to be between -1 and 1, and this opertion is reversed for the output so the predicted price appears in the appropriate scale.

The data that was used in this study was obtained from Goldman Sachs Group (GS), Inc. and Microsoft Corp. (MSFT) . The data covers 734 trading days (4 January 2010 to 31 December 2012). Each instance of the data consisted of daily statistics: low price, high price, opening price, close price, and trading volume. To facilitate the training and testing of the model, this data was split into training data and test data with 80% and 20% of the original data, respectively. In addition to the daily-statistics variables in the data, six more variables were created to reflect financial indicators.

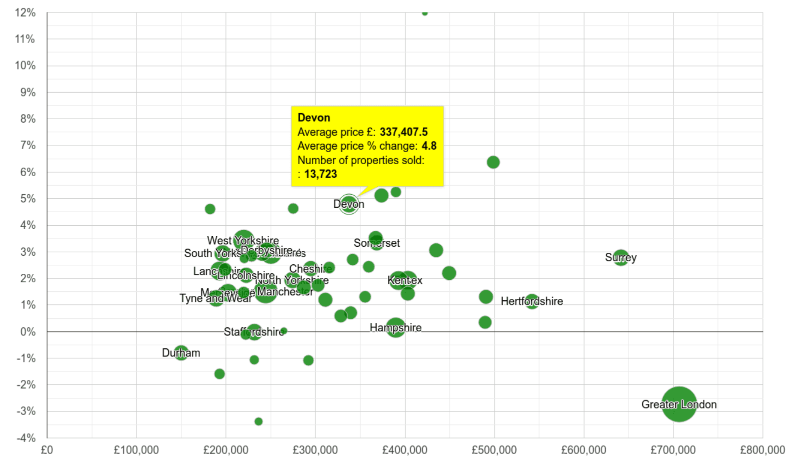
The performance of the model were evaluated using mean absolute percentage error (MAPE) performance metric. MAPE was calculated using this formula:

$$MAPE=\frac{\sum_{i=1}^{r}(abs(y_i-p_i)/y_i)}{r}\times 100$$

where p_i
is the predicted stock price on day i
, y_i
is the actual stock price on day i

, and r
is the number of trading days.

When applied on the test data, The model achieved a MAPE score of 1.0561 for MSFT part, and 1.3291 for GS part. Figure 1 shows the actual values and predicted values for both GS and MSFT data.



ne studies related to our work from scientific journals. In section 3, we go through data preparation including data cleaning, outlier removal, and feature engineering. Next in section 4, we discuss the type of our problem and the type of machine-learning prediction that should be applied; we also list the prediction techniques that will be used. In section 5, we choose algorithms to implement the techniques in section 4; we build models based on these algorithms; we also train and test each model. In section 6, we analyze and compare the results we got from section 5 and conclude the paper.

Literature Review

In this section, we look at five recent studies that are related to our topic and see how models were built and what results were achieved in these studies.

Stock Market Prediction Using Bayesian-Regularized Neural Networks

In a study done by Ticknor (2013), he used Bayesian regularized artificial neural network to predict the future operation of financial market. Specifically, he built a model to predict future stock prices. The input of the model is previous stock statistics in addition to some financial technical data. The output of the model is the next-day closing price of the corresponding stocks.

The model proposed in the study is built using Bayesian regularized neural network. The weights of this type of networks are given a probabilistic nature. This allows the network to penalize very complex models (with many hidden layers) in an automatic manner. This in turn will reduce the overfitting of the model.

The model consists of a feedforward neural network which has three layers: an input layer, one hidden layer, and an output layer. The author chose the number of neurons in the hidden layer based on experimental methods.The input data of the model is normalized to be between -1 and 1, and this opertion is reversed for the output so the predicted price appears in the appropriate scale.

The data that was used in this study was obtained from Goldman Sachs Group (GS), Inc. and Microsoft Corp. (MSFT) . The data covers 734 trading days (4 January 2010 to 31 December 2012). Each instance of the data consisted of daily statistics: low price, high price, opening price, close price, and trading volume. To facilitate the training and testing of the model, this data was split into training data and test data with 80% and 20% of the original data, respectively. In addition to the daily-statistics variables in the data, six more variables were created to reflect financial indicators.

The performance of the model were evaluated using mean absolute percentage error (MAPE) performance metric. MAPE was calculated using this formula:

MAPE=∑ri=1(abs(yi−pi)/yi)r×100
where pi
 is the predicted stock price on day i
, yi
 is the actual stock price on day i
, and r
 is the number of trading days.

When applied on the test data, The model achieved a MAPE score of 1.0561 for MSFT part, and 1.3291 for GS part. Figure 1 shows the actual values and predicted values for both GS and MSFT data.

Figure 1: Predicted vs. actual price

Stock Market Prediction Using A Machine Learning Model

In another study done by Hegazy, Soliman, and Salam (2014), a system was proposed to predict daily stock market prices. The system combines particle swarm optimization (PSO) and least square support vector machine (LS-SVM), where PSO was used to optimize LV-SVM.

The authors claim that in most cases, artificial neural networks (ANNs) are subject to the overfitting problem. They state that support vector machines algorithm (SVM) was developed as an alternative that doesn't suffer from overfitting. They attribute this advantage to SVMs being based on the solid foundations of VC-theory. They further elaborate that LS-SVM method was reformulation of traditional SVM method that uses a regularized least squares function with equality constraints to obtain a linear system that satisfies Karush-Kuhn-Tucker conditions for getting an optimal solution.

The authors describe PSO as a popular evolutionary optimization method that was inspired by organism social behavior like bird flocking. They used it to find the optimal parameters for LS-SVM. These parameters are the cost penalty C
, kernel parameter γ
, and insensitive loss function ϵ
.

The model proposed in the study was based on the analysis of historical data and technical financial indicators and using LS-SVM optimized by PSO to predict future daily stock prices. The model input was six vectors representing the historical data and the technical financial indicators. The model output was the future price. The model used is represented in Figure 2.

Figure 2: The structure of the model used

Regarding the technical financial indicators, five were derived from the raw data: relative strength index (RSI), money flow index (MFI), exponential moving average (EMA), stochastic oscillator (SO), and moving average convergence/divergence (MACD). These indicators are known in the domain of stock market.

The model was trained and tested using datasets taken from https://finance.yahoo.com/. The datasets were from Jan 2009 to Jan 2012 and include stock data for many companies like Adobe and HP. All datasets were partitioned into a training set with 70% of the data and a test set with 30% of the data. Three models were trained and tested: LS-SVM-PSO model, LS-SVM model, and ANN model. The results obtained in the study showed that LS-SVM-PSO model had the best performance. Figure 3 shows a comparison between the mean square error (MSE) of the three models for the stocks of many companies.

Figure 3: MSE comparison

House Price Prediction Using Multilevel Model and Neural Networks
A different study was done by Feng and Jones (2015) to preduct house prices. Two models were built: a multilevel model (MLM) and an artificial neural network model (ANN). These two models were compared to each other and to a hedonic price model (HPM).

The multilevel model integrates the micro-level that specifies the relationships between houses within a given neighbourhood, and the macro-level equation which specifies the relationships between neighbouhoods. The hedonic price model is a model that estimates house prices using some attributes such as the number of bedrooms in the house, the size of the house, etc.

The data used in the study contains house prices in Greater Bristol area between 2001 and 2013. Secondary data was obtained from the Land Registry, the Population Census and

Neighbourhood Statistics to be used in order to make the models suitable for national usage. The authors listed many reasons on why they chose the Greater Bristol area such as its diverse urban and rural blend and its different property types. Each record in the dataset contains data about a house in the area: it contains the address, the unit postcode, property type, the duration (freehold or leasehold), the sale price, the date of the sale, and whether the house was newly-built when it was sold. In total, the dataset contains around 65,000 entries. To enable model training and testing, the dataset was divided into a training set that contains data about house sales from 2001 to 2012, and a test set that contains data about house sales in 2013.

The three models (MLM, ANN, and HPM) were tested using three senarios. In the first senario, locational and measured neighbourhood attributes were not included in the data. In the second senario, grid references of house location were included in the data. In the third senario, measured neighbourhood attributes were included in the data. The models were compared in goodness of fit where R2
 was the metric, predictive accuracy where mean absolute error (MAE) and mean absolute percentage error (MAPE) were the metrics, and explanatory power. HPM and MLM models were fitted using MLwiN software, and ANN were fitted using IBM SPSS software. Figure 4 shows the performance of each model regarding fit goodness and predictive accuracy. It shows that MLM model has better performance in general than other models.

Figure 4: Model performance comparison

Composition of Models and Feature Engineering to Win Algorithmic Trading Challenge
A study done by de Abril and Sugiyama (2013) introduced the techniques and ideas used to win Algorithmic Trading Challenge, a competition held on Kaggle. The goal of the competition was to develop a model that can predict the short-term response of order-driven markets after a big liquidity shock. A liquidity shock happens when a trade or a sequence of trades causes an acute shortage of liquidity (cash for example).

The challenge data contains a training dataset and a test dataset. The training dataset has around 754,000 records of trade and quote observations for many securities of London Stock Exchange before and after a liquidity shock. A trade event happens when shares are sold or bought, whereas a quote event happens when the ask price or the best bid changes.

A separate model was built for bid and another for ask. Each one of these models consists of K random-forest sub-models. The models predict the price at a particular future time.

The authors spent much effort on feature engineering. They created more than 150 features. These features belong to four categories: price features, liquidity-book features, spread features (bid/ask spread), and rate features (arrival rate of orders/quotes). They applied a feature selection algorithm to obtain the optimal feature set (Fb
) for bid sub-models and the optimal feature set (Fa
) of all ask sub-models. The algorithm applied eliminates features in a backward manner in order to get a feature set with reasonable computing time and resources.

Three instances of the final model proposed in the study were trained on three datasets; each one of them consists of 50,000 samples sampled randomly from the training dataset. Then, the three models were applied to the test dataset. The predictions of the three models were then averaged to obtain the final prediction. The proposed method achieved a RMSE score of 0.77 approximately.

Using K-Nearest Neighbours for Stock Price Prediction

Alkhatib, Najadat, Hmeidi, and Shatnawi (2013) have done a study where they used the k-nearest neighbours (KNN) algorithm to predict stock prices. In this study, they expressed the stock prediction problem as a similarity-based classification, and they represented the historical stock data as well as test data by vectors.

The authors listed the steps of predicting the closing price of stock market using KNN as follows:

The number of neaerest neighbours is chosen
The distance between the new record and the training data is computed
Training data is sorted according to the calculated distance
Majority voting is applied to the classes of the k nearest neighbours to determine the predicted value of the new record
The data used in the study is stock data of five companies listed on the Jordanian stock exchange. The data range is from 4 June 2009 to 24 December 2009. Each of the five companies has around 200 records in the data. Each record has three variables: closing price, low price, and high price. The author stated that the closing price is the most important feature in determining the prediction value of a stock using KNN.

After applying KNN algorithm, the authors summarized the prediction performance evaluation using different metrics in a the table show

Reading the Dataset

The first step is reading the dataset from the csv file we downloaded. We will use the read_csv() function from Pandas Python package:

```
import pandas as pd
import numpy as np

dataset = pd.read_csv("../input/AmesHousing.csv")
```

Getting A Feel of the Dataset
Let's display the first few rows of the dataset to get a feel of it:

```
# Configuring float numbers format
pd.options.display.float_format = '{:20.2f}'.format
dataset.head(n=5)
Order	PID	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Alley	Lot
Shape	Land Contour	Utilities	Lot Config	Land Slope	Neighborhood	Condition 1
Condition 2	Bldg Type	House Style	Overall Qual	Overall Cond	Year Built	Year
Remod/Add	Roof Style	Roof Matl	Exterior 1st	Exterior 2nd	Mas Vnr Type	Mas
Vnr Area	Exter Qual	Exter Cond	Foundation	Bsmt Qual	Bsmt Cond	Bsmt
Exposure	BsmtFin Type 1	BsmtFin SF 1	BsmtFin Type 2	BsmtFin SF 2	Bsmt
Unf SF	Total Bsmt SF	...	Central Air	Electrical	1st Flr SF	2nd Flr SF	Low
Qual Fin SF	Gr Liv Area	Bsmt Full Bath	Bsmt Half Bath	Full Bath	Half Bath
Bedroom AbvGr	Kitchen AbvGr	Kitchen Qual	TotRms AbvGrd	Functional
Fireplaces	Fireplace Qu	Garage Type	Garage Yr Blt	Garage Finish	Garage Cars
Garage Area	Garage Qual	Garage Cond	Paved Drive	Wood Deck SF	Open Porch
SF	Enclosed Porch	3Ssn Porch	Screen Porch	Pool Area	Pool QC	Fence
Misc Feature	Misc Val	Mo Sold	Yr Sold	Sale Type	Sale Condition
SalePrice
0	1	526301100	20	RL	141.00	31770	Pave	NaN	IR1	Lvl	AllPub
Corner	Gtl	NAmes	Norm	Norm	1Fam	1Story	6	5	1960	1960	Hip
CompShg	BrkFace	Plywood	Stone	112.00	TA	TA	CBlock	TA	Gd
Gd	BLQ	639.00	Unf	0.00	441.00	1080.00	...	Y	SBrkr	1656	0
0	1656	1.00	0.00	1	0	3	1	TA	7	Typ	2	Gd
Attchd	1960.00	Fin	2.00	528.00	TA	TA	P	210	62	0	0
0	0	NaN	NaN	NaN	0	5	2010	WD	Normal	215000
1	2	526350040	20	RH	80.00	11622	Pave	NaN	Reg	Lvl	AllPub
Inside	Gtl	NAmes	Feedr	Norm	1Fam	1Story	5	6	1961	1961	Gable
CompShg	VinylSd	VinylSd	None	0.00	TA	TA	CBlock	TA	TA
No	Rec	468.00	LwQ	144.00	270.00	882.00	...	Y	SBrkr	896	0	0
896	0.00	0.00	1	0	2	1	TA	5	Typ	0	NaN	Attchd
1961.00	Unf	1.00	730.00	TA	TA	Y	140	0	0	0	120
0	NaN	MnPrv	NaN	0	6	2010	WD	Normal	105000
2	3	526351010	20	RL	81.00	14267	Pave	NaN	IR1	Lvl	AllPub
Corner	Gtl	NAmes	Norm	Norm	1Fam	1Story	6	6	1958	1958	Hip
CompShg	Wd Sdng	Wd Sdng	BrkFace	108.00	TA	TA	CBlock	TA
TA	No	ALQ	923.00	Unf	0.00	406.00	1329.00	...	Y	SBrkr	1329
0	0	1329	0.00	0.00	1	1	3	1	Gd	6	Typ	0
NaN	Attchd	1958.00	Unf	1.00	312.00	TA	TA	Y	393	36	0
0	0	0	NaN	NaN	Gar2	12500	6	2010	WD	Normal	172000
3	4	526353030	20	RL	93.00	11160	Pave	NaN	Reg	Lvl	AllPub
Corner	Gtl	NAmes	Norm	Norm	1Fam	1Story	7	5	1968	1968	Hip
CompShg	BrkFace	BrkFace	None	0.00	Gd	TA	CBlock	TA	TA
```

```
No	ALQ	1065.00		Unf	0.00	1045.00		2110.00		...	Y	SBrkr
2110	0	0	2110	1.00	0.00	2	1	3	1	Ex	8	Typ
2	TA	Attchd	1968.00		Fin	2.00	522.00	TA	TA	Y	0	0
0	0	0	0	NaN	NaN	NaN	0	4	2010	WD	Normal
244000
4	5	527105010	60	RL	74.00	13830	Pave	NaN	IR1	Lvl	AllPub
Inside	Gtl	Gilbert	Norm	Norm	1Fam	2Story	5	5	1997	1998	Gable
CompShg		VinylSd		VinylSd		None	0.00	TA	TA	PConc	Gd	TA
No	GLQ	791.00	Unf	0.00	137.00	928.00	...	Y	SBrkr	928	701	0
1629	0.00	0.00	2	1	3	1	TA	6	Typ	1	TA	Attchd
1997.00		Fin	2.00	482.00	TA	TA	Y	212	34	0	0	0
0	NaN	MnPrv	NaN	0	3	2010	WD	Normal	189900
```

Now, let's get statistical information about the numeric columns in our dataset. We want to know the mean, the standard deviation, the minimum, the maximum, and the 50th percentile (the median) for each numeric column in the dataset:

```
dataset.describe(include=[np.number], percentiles=[.5]) \
    .transpose().drop("count", axis=1)
```

```
mean	std	min	50%	max
Order	1465.50		845.96	1.00	1465.50		2930.00
PID	714464496.99	188730844.65	526301100.00	535453620.00	1007100110.00
MS SubClass	57.39	42.64	20.00	50.00	190.00
Lot Frontage	69.22	23.37	21.00	68.00	313.00
Lot Area	10147.92		7880.02		1300.00		9436.50		215245.00
Overall Qual	6.09	1.41	1.00	6.00	10.00
Overall Cond	5.56	1.11	1.00	5.00	9.00
Year Built	1971.36		30.25	1872.00		1973.00		2010.00
Year Remod/Add	1984.27		20.86	1950.00		1993.00		2010.00
Mas Vnr Area	101.90	179.11	0.00	0.00	1600.00
BsmtFin SF 1	442.63	455.59	0.00	370.00	5644.00
BsmtFin SF 2	49.72	169.17	0.00	0.00	1526.00
Bsmt Unf SF	559.26	439.49	0.00	466.00	2336.00
Total Bsmt SF	1051.61		440.62	0.00	990.00	6110.00
1st Flr SF	1159.56		391.89	334.00	1084.00		5095.00
2nd Flr SF	335.46	428.40	0.00	0.00	2065.00
Low Qual Fin SF	4.68	46.31	0.00	0.00	1064.00
Gr Liv Area	1499.69		505.51	334.00	1442.00		5642.00
Bsmt Full Bath	0.43	0.52	0.00	0.00	3.00
Bsmt Half Bath	0.06	0.25	0.00	0.00	2.00
Full Bath	1.57	0.55	0.00	2.00	4.00
Half Bath	0.38	0.50	0.00	0.00	2.00
Bedroom AbvGr	2.85	0.83	0.00	3.00	8.00
Kitchen AbvGr	1.04	0.21	0.00	1.00	3.00
TotRms AbvGrd	6.44	1.57	2.00	6.00	15.00
```

```
Fireplaces	0.60	0.65	0.00	1.00	4.00
Garage Yr Blt	1978.13		25.53	1895.00		1979.00		2207.00
Garage Cars	1.77	0.76	0.00	2.00	5.00
Garage Area	472.82	215.05	0.00	480.00	1488.00
Wood Deck SF		93.75	126.36	0.00	0.00	1424.00
Open Porch SF		47.53	67.48	0.00	27.00	742.00
Enclosed Porch		23.01	64.14	0.00	0.00	1012.00
3Ssn Porch	2.59	25.14	0.00	0.00	508.00
Screen Porch	16.00	56.09	0.00	0.00	576.00
Pool Area	2.24	35.60	0.00	0.00	800.00
Misc Val	50.64	566.34	0.00	0.00	17000.00
Mo Sold		6.22	2.71	1.00	6.00	12.00
Yr Sold		2007.79		1.32	2006.00		2008.00		2010.00
SalePrice	180796.06	79886.69	12789.00	160000.00	755000.00
```

From the table above, we can see, for example, that the average lot area of the houses in our dataset is 10,147.92 ft2 with a standard deviation of 7,880.02 ft2. We can see also that the minimum lot area is 1,300 ft2 and the maximum lot area is 215,245 ft2 with a median of 9,436.5 ft2. Similarly, we can get a lot of information about our dataset variables from the table.

Then, we move to see statistical information about the non-numerical columns in our dataset:

```
dataset.describe(include=[np.object]).transpose() \
    .drop("count", axis=1)
```

```
unique top	freq
MS Zoning	7	RL	2273
Street	2	Pave	2918
Alley	2	Grvl	120
Lot Shape	4	Reg	1859
Land Contour	4	Lvl	2633
Utilities	3	AllPub	2927
Lot Config	5	Inside	2140
Land Slope	3	Gtl	2789
Neighborhood	28	NAmes		443
Condition 1	9	Norm	2522
Condition 2	8	Norm	2900
Bldg Type	5	1Fam	2425
House Style	8	1Story	1481
Roof Style	6	Gable	2321
Roof Matl	8	CompShg		2887
Exterior 1st	16	VinylSd		1026
Exterior 2nd	17	VinylSd		1015
Mas Vnr Type	5	None	1752
Exter Qual	4	TA	1799
Exter Cond	5	TA	2549
```

Foundation	6	PConc	1310
Bsmt Qual	5	TA	1283
Bsmt Cond	5	TA	2616
Bsmt Exposure	4	No	1906
BsmtFin Type 1	6	GLQ	859
BsmtFin Type 2	6	Unf	2499
Heating	6	GasA	2885
Heating QC	5	Ex	1495
Central Air	2	Y	2734
Electrical	5	SBrkr	2682
Kitchen Qual	5	TA	1494
Functional	8	Typ	2728
Fireplace Qu	5	Gd	744
Garage Type	6	Attchd	1731
Garage Finish	3	Unf	1231
Garage Qual	5	TA	2615
Garage Cond	5	TA	2665
Paved Drive	3	Y	2652
Pool QC	4	Ex	4
Fence	4	MnPrv	330
Misc Feature	5	Shed	95
Sale Type	10	WD	2536
Sale Condition	6	Normal	2413

In the table we got, count represents the number of non-null values in each column, unique represents the number of unique values, top represents the most frequent element, and freq represents the frequency of the most frequent element.

Data Cleaning

Dealing with Missing Values

We should deal with the problem of missing values because some machine learning models don't accept data with missing values. Firstly, let's see the number of missing values in our dataset. We want to see the number and the percentage of missing values for each column that actually contains missing values.

```
# Getting the number of missing values in each column
num_missing = dataset.isna().sum()
# Excluding columns that contains 0 missing values
num_missing = num_missing[num_missing > 0]
# Getting the percentages of missing values
percent_missing = num_missing * 100 / dataset.shape[0]
# Concatenating the number and perecentage of missing values
# into one dataframe and sorting it
pd.concat([num_missing, percent_missing], axis=1,
        keys=['Missing Values', 'Percentage']).\
        sort_values(by="Missing Values", ascending=False)
Missing Values          Percentage
Pool QC         2917    99.56
Misc Feature    2824    96.38
Alley   2732    93.24
Fence   2358    80.48
Fireplace Qu    1422    48.53
Lot Frontage    490     16.72
Garage Cond     159     5.43
Garage Qual     159     5.43
Garage Finish   159     5.43
Garage Yr Blt   159     5.43
Garage Type     157     5.36
Bsmt Exposure           83      2.83
BsmtFin Type 2          81      2.76
BsmtFin Type 1          80      2.73
Bsmt Qual       80      2.73
Bsmt Cond       80      2.73
Mas Vnr Area    23      0.78
Mas Vnr Type    23      0.78
Bsmt Half Bath          2       0.07
Bsmt Full Bath2         0.07
Total Bsmt SF   1       0.03
Bsmt Unf SF     1       0.03
Garage Cars     1       0.03
Garage Area     1       0.03
BsmtFin SF 2    1       0.03
BsmtFin SF 1    1       0.03
Electrical      1       0.03
```

Now we start dealing with these missing values.

Pool QC

The percentage of missing values in Pool QC column is 99.56% which is very high. We think that a missing value in this column denotes that the corresponding house doesn't have a pool. To verify this, let's take a look at the values of Pool Area column:

```
dataset["Pool Area"].value_counts()
0     2917.
```

We can see that there are 2917 entries in Pool Area column that have a value of 0. This verfies our hypothesis that each house without a pool has a missing value in Pool QC column and a value of 0 in Pool Area column. So let's fill the missing values in Pool QC column with "No Pool":

```
dataset["Pool QC"].fillna("No Pool", inplace=True)
```

Misc Feature

The percentage of missing values in Pool QC column is 96.38% which is very high also. Let's take a look at the values of Misc Val column:

```
dataset["Misc Val"].value_counts()
0        2827
400        18
500        13
450         9
600         8
700         7
2000        7
650         3
1200        3
1500        3
4500        2
2500        2
480         2
3000        2
12500       1
300         1
350         1
8300        1
420         1
80          1
54          1
460         1
490         1
3500        1
560         1
17000       1
15500       1
750         1
800         1
900         1
1000        1
1150        1
1300        1
1400        1
1512        1
```

```
6500     1
455      1
620      1
Name: Misc Val, dtype: int64
```

We can see that Misc Val column has 2827 entries with a value of 0. Misc Feature has 2824 missing values. Then, as with Pool QC, we can say that each house without a "miscellaneous feature" has a missing value in Misc Feature column and a value of 0 in Misc Val column. So let's fill the missing values in Misc Feature column with "No Feature":

```
dataset['Misc Feature'].fillna('No feature', inplace=True)
```

Alley, Fence, and Fireplace Qu

According to the dataset documentation, NA in Alley, Fence, and Fireplace Qu columns denotes that the house doesn't have an alley, fence, or fireplace. So we fill in the missing values in these columns with "No Alley", "No Fence", and "No Fireplace" accordingly:

```
dataset['Alley'].fillna('No Alley', inplace=True)
dataset['Fence'].fillna('No Fence', inplace=True)
dataset['Fireplace Qu'].fillna('No Fireplace', inplace=True)
```

Lot Frontage

As we saw previously, Lot Frontage represents the linear feet of street connected to the house. So we assume that the missing values in this column indicates that the house is not connected to any street, and we fill in the missing values with 0:

```
dataset['Lot Frontage'].fillna(0, inplace=True)
```

Garage Cond, Garage Qual, Garage Finish, Garage Yr Blt, Garage Type, Garage Cars, and Garage Area

According to the dataset documentation, NA in Garage Cond, Garage Qual, Garage Finish, and Garage Type indicates that there is no garage in the house. So we fill in the missing values in these columns with "No Garage". We notice that Garage Cond, Garage Qual, Garage Finish, Garage Yr Blt columns have 159 missing values, but Garage Type has 157 and both Garage Cars and Garage Area have one missing value. Let's take a look at the row that contains the missing value in Garage Cars:

```
garage_columns = [col for col in dataset.columns if col.startswith("Garage")]
dataset[dataset['Garage Cars'].isna()][garage_columns]
Garage Type   Garage Yr Blt  Garage Finish Garage Cars   Garage Area   Garage Qual
Garage Cond
2236    Detchd nan     NaN    nan     nan     NaN    NaN
```

We can see that this is the same row that contains the missing value in Garage Area, and that all garage columns except Garage Type are null in this row, so we will fill the missing values in Garage Cars and Garage Area with 0.

We saw that there are 2 rows where Garage Type is not null while Garage Cond, Garage Qual, Garage Finish, and Garage Yr Blt columns are null. Let's take a look at these two rows:

```
dataset[~pd.isna(dataset['Garage Type']) &
       pd.isna(dataset['Garage Qual'])][garage_columns]
Garage Type   Garage Yr Blt  Garage Finish Garage Cars   Garage Area   Garage Qual
Garage Cond
1356    Detchd nan     NaN     1.00     360.00 NaN     NaN
2236    Detchd nan     NaN     nan      nan     NaN     NaN
```

We will replace the values of Garage Type with "No Garage" in these two rows also.

For Garage Yr Blt, we will fill in missing values with 0 since this is a numerical column:

```
dataset['Garage Cars'].fillna(0, inplace=True)
dataset['Garage Area'].fillna(0, inplace=True)

dataset.loc[~pd.isna(dataset['Garage Type']) &
          pd.isna(dataset['Garage Qual']), "Garage Type"] = "No Garage"

for col in ['Garage Type', 'Garage Finish', 'Garage Qual', 'Garage Cond']:
   dataset[col].fillna('No Garage', inplace=True)

dataset['Garage Yr Blt'].fillna(0, inplace=True)
```

Bsmt Exposure, BsmtFin Type 2, BsmtFin Type 1, Bsmt Qual, Bsmt Cond, Bsmt Half Bath, Bsmt Full Bath, Total Bsmt SF, Bsmt Unf SF, BsmtFin SF 2, and BsmtFin SF 1

According to the dataset documentation, NA in any of the first five of these columns indicates that there is no basement in the house. So we fill in the missing values in these columns with "No Basement". We notice that the first five of these columns have 80 missing values, but BsmtFin Type 2 has 81, Bsmt Exposure has 83, Bsmt Half Bath and Bsmt Full Bath each has 2, and each of the others has 1. Let's take a look at the rows where Bsmt Half Bath is null:

```
bsmt_columns = [col for col in dataset.columns if "Bsmt" in col]
dataset[dataset['Bsmt Half Bath'].isna()][bsmt_columns]
Bsmt Qual      Bsmt Cond      Bsmt Exposure          BsmtFin Type 1          BsmtFin SF 1
BsmtFin Type 2          BsmtFin SF 2  Bsmt Unf SF   Total Bsmt SF Bsmt Full BathBsmt Half
Bath
1341    NaN     NaN     NaN     NaN     nan      NaN     nan      nan      nan      nan      nan
1497    NaN     NaN     NaN     NaN     0.00     NaN     0.00     0.00     0.00     nan      nan
```

We can see that these are the same rows that contain the missing values in Bsmt Full Bath, and that one of these two rows is contains the missing value in each of Total Bsmt SF, Bsmt Unf SF, BsmtFin SF 2, and BsmtFin SF 1 columns. We notice also that Bsmt Exposure, BsmtFin Type 2, BsmtFin Type 1, Bsmt Qual, and Bsmt Cond are null in these rows, so we will fill the missing values in Bsmt Half Bath, Bsmt Full Bath, Total Bsmt SF, Bsmt Unf SF, BsmtFin SF 2, and BsmtFin SF 1 columns with 0.

We saw that there are 3 rows where Bsmt Exposure is null while BsmtFin Type 1, Bsmt Qual, and Bsmt Cond are not null. Let's take a look at these three rows:

```
dataset[~pd.isna(dataset['Bsmt Cond']) &
       pd.isna(dataset['Bsmt Exposure'])][bsmt_columns]
Bsmt Qual	Bsmt Cond	Bsmt Exposure	BsmtFin Type 1	BsmtFin SF 1	
BsmtFin Type 2	BsmtFin SF 2	Bsmt Unf SF	Total Bsmt SF	Bsmt Full Bath	Bsmt Half 
Bath
66	Gd	TA	NaN	Unf	0.00	Unf	0.00	1595.00	1595.00	0.00	
0.00
1796	Gd	TA	NaN	Unf	0.00	Unf	0.00	725.00	725.00	0.00	0.00
2779	Gd	TA	NaN	Unf	0.00	Unf	0.00	936.00	936.00	0.00	0.00
```

We will fill in the missing values in Bsmt Exposure for these three rows with "No". According to the dataset documentation, "No" for Bsmt Exposure means "No Exposure":

Let's now take a look at the row where BsmtFin Type 2 is null while BsmtFin Type 1, Bsmt Qual, and Bsmt Cond are not null:

```
dataset[~pd.isna(dataset['Bsmt Cond']) &
       pd.isna(dataset['BsmtFin Type 2'])][bsmt_columns]
Bsmt Qual	Bsmt Cond	Bsmt Exposure	BsmtFin Type 1	BsmtFin SF 1	
BsmtFin Type 2	BsmtFin SF 2	Bsmt Unf SF	Total Bsmt SF	Bsmt Full Bath	Bsmt Half 
Bath
444	Gd	TA	No	GLQ	1124.00		NaN	479.00	1603.00		3206.00	
1.00	0.00
```

We will fill in the missing value in BsmtFin Type 2 for this row with "Unf". According to the dataset documentation, "Unf" for BsmtFin Type 2 means "Unfinished":

```
for col in ["Bsmt Half Bath", "Bsmt Full Bath", "Total Bsmt SF",
           "Bsmt Unf SF", "BsmtFin SF 2", "BsmtFin SF 1"]:
    dataset[col].fillna(0, inplace=True)

dataset.loc[~pd.isna(dataset['Bsmt Cond']) &
           pd.isna(dataset['Bsmt Exposure']), "Bsmt Exposure"] = "No"
dataset.loc[~pd.isna(dataset['Bsmt Cond']) &
           pd.isna(dataset['BsmtFin Type 2']), "BsmtFin Type 2"] = "Unf"

for col in ["Bsmt Exposure", "BsmtFin Type 2",
           "BsmtFin Type 1", "Bsmt Qual", "Bsmt Cond"]:
    dataset[col].fillna("No Basement", inplace=True)
```

Mas Vnr Area and Mas Vnr Type

Each of these two columns have 23 missing values. We will fill in these missing values with "None" for Mas Vnr Type and with 0 for Mas Vnr Area. We use "None" for Mas Vnr Type because in the dataset documentation, "None" for Mas Vnr Type means "None" (i.e. no masonry veneer):

```
dataset['Mas Vnr Area'].fillna(0, inplace=True)
dataset['Mas Vnr Type'].fillna("None", inplace=True)
```

Electrical
This column has one missing value. We will fill in this value with the mode of this column: