House prices data

Bruna Wundervald

October, 2018

```
# Packages and models import
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.svm import SVR
from pandas.api.types import is_object_dtype
import csv
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
```

Data fields

Here's a brief version of what you'll find in the data description file.

- SalePrice - the property's sale price in dollars (target variable).
- MSSubClass: The building class
- MSZoning: The general zoning classification
- LotFrontage: Linear feet of street connected to property
- LotArea: Lot size in square feet
- Street: Type of road access
- Alley: Type of alley access
- LotShape: General shape of property
- LandContour: Flatness of the property
- Utilities: Type of utilities available
- LotConfig: Lot configuration
- LandSlope: Slope of property
- Neighborhood: Physical locations within Ames city limits
- Condition1: Proximity to main road or railroad
- Condition2: Proximity to main road or railroad (if a second is present)
- BldgType: Type of dwelling
- HouseStyle: Style of dwelling
- OverallQual: Overall material and finish quality
- OverallCond: Overall condition rating
- YearBuilt: Original construction date
- YearRemodAdd: Remodel date
- RoofStyle: Type of roof
- RoofMatl: Roof material
- Exterior1st: Exterior covering on house
- Exterior2nd: Exterior covering on house (if more than one material)
- MasVnrType: Masonry veneer type
- MasVnrArea: Masonry veneer area in square feet
- ExterQual: Exterior material quality

- ExterCond: Present condition of the material on the exterior
- Foundation: Type of foundation
- BsmtQual: Height of the basement
- BsmtCond: General condition of the basement
- BsmtExposure: Walkout or garden level basement walls
- BsmtFinType1: Quality of basement finished area
- BsmtFinSF1: Type 1 finished square feet
- BsmtFinType2: Quality of second finished area (if present)
- BsmtFinSF2: Type 2 finished square feet
- BsmtUnfSF: Unfinished square feet of basement area
- TotalBsmtSF: Total square feet of basement area
- Heating: Type of heating
- HeatingQC: Heating quality and condition
- CentralAir: Central air conditioning
- Electrical: Electrical system
- 1stFlrSF: First Floor square feet
- 2ndFlrSF: Second floor square feet
- LowQualFinSF: Low quality finished square feet (all floors)
- GrLivArea: Above grade (ground) living area square feet
- BsmtFullBath: Basement full bathrooms
- BsmtHalfBath: Basement half bathrooms
- FullBath: Full bathrooms above grade
- HalfBath: Half baths above grade
- Bedroom: Number of bedrooms above basement level
- Kitchen: Number of kitchens
- KitchenQual: Kitchen quality
- TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
- Functional: Home functionality rating
- Fireplaces: Number of fireplaces
- FireplaceQu: Fireplace quality
- GarageType: Garage location
- GarageYrBlt: Year garage was built
- GarageFinish: Interior finish of the garage
- GarageCars: Size of garage in car capacity
- GarageArea: Size of garage in square feet
- GarageQual: Garage quality
- GarageCond: Garage condition
- PavedDrive: Paved driveway
- WoodDeckSF: Wood deck area in square feet
- OpenPorchSF: Open porch area in square feet
- EnclosedPorch: Enclosed porch area in square feet
- 3SsnPorch: Three season porch area in square feet
- ScreenPorch: Screen porch area in square feet
- PoolArea: Pool area in square feet
- PoolQC: Pool quality
- Fence: Fence quality
- MiscFeature: Miscellaneous feature not covered in other categories
- MiscVal: $Value of miscellaneous feature
- MoSold: Month Sold
- YrSold: Year Sold
- SaleType: Type of sale
- SaleCondition: Condition of sale

```python
# Reading data
da =  pd.read_csv('data/house-prices/train.csv')
da.iloc[0]

# Replacing NaNs with 0
```

```
## Id                   1
## MSSubClass          60
## MSZoning            RL
## LotFrontage         65
## LotArea           8450
## Street            Pave
## Alley              NaN
## LotShape           Reg
## LandContour        Lvl
## Utilities       AllPub
## LotConfig       Inside
## LandSlope          Gtl
## Neighborhood   CollgCr
## Condition1        Norm
## Condition2        Norm
## BldgType          1Fam
## HouseStyle      2Story
## OverallQual          7
## OverallCond          5
## YearBuilt         2003
## YearRemodAdd      2003
## RoofStyle        Gable
## RoofMatl       CompShg
## Exterior1st    VinylSd
## Exterior2nd    VinylSd
## MasVnrType     BrkFace
## MasVnrArea         196
## ExterQual           Gd
## ExterCond           TA
## Foundation       PConc
##                 ...
## BedroomAbvGr         3
## KitchenAbvGr         1
## KitchenQual         Gd
## TotRmsAbvGrd         8
## Functional         Typ
## Fireplaces           0
## FireplaceQu        NaN
## GarageType      Attchd
## GarageYrBlt       2003
## GarageFinish       RFn
## GarageCars           2
## GarageArea         548
## GarageQual          TA
## GarageCond          TA
## PavedDrive           Y
## WoodDeckSF           0
## OpenPorchSF         61
```

```
## EnclosedPorch          0
## 3SsnPorch              0
## ScreenPorch            0
## PoolArea               0
## PoolQC               NaN
## Fence                NaN
## MiscFeature          NaN
## MiscVal                0
## MoSold                 2
## YrSold              2008
## SaleType              WD
## SaleCondition     Normal
## SalePrice         208500
## Name: 0, Length: 81, dtype: object
```

```
da.fillna(0, inplace = True)
da.shape
```

```
## (1460, 81)
```

The plots above show the density of the response variable and the dentsity of its log transformation. Logarithmically transforming variables in a regression model is a very convenient way of transforming a highly skewed variable (usually the response) into one that is more approximately Normal. When we look at the plots, is easy to see that the original variable is skewed, as it has some very high values, seeing that we are dealing with house prices. Comparing with the transformed variable, we can now see that this one looks way closer to a Normal distribution, what justifies the transformation.

```
# Density plot of y  and y in log scale -----------------------------------
plt.clf() # clean plot environment x
plt.figure(1)
```

```
plt.figure(figsize = (14, 10))
```

```
plt.subplot(211)
```

```
sns.distplot(da['SalePrice'], bins = 30, kde = True, color = 'tomato', rug = True)
```

```
plt.xlabel('Price', fontsize = 17)
```

```
plt.ylabel('Histogram and density', fontsize = 17)
```

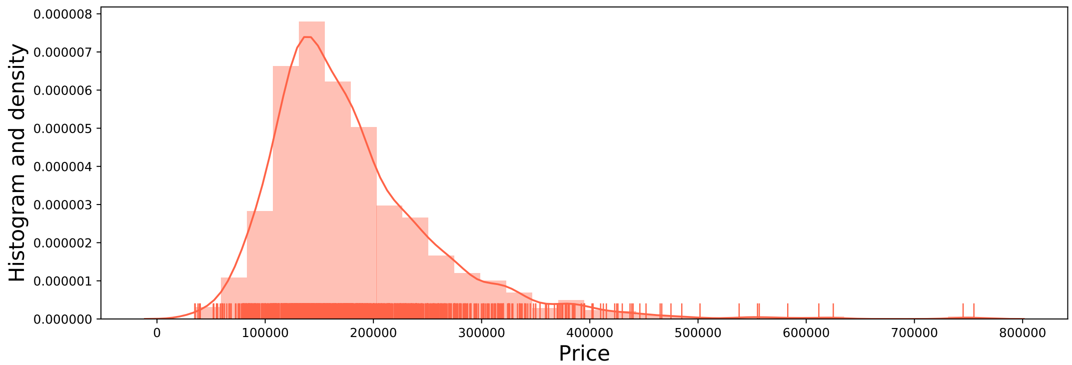
```
plt.subplot(212)
```

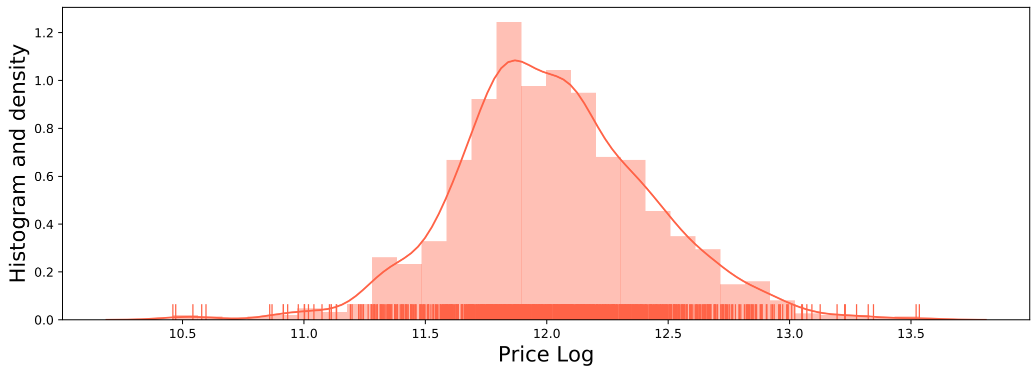
```
sns.distplot(np.log(da['SalePrice']), bins = 30, kde = True, color = 'tomato', rug = True)
```

```
plt.xlabel('Price Log', fontsize = 17)
```

```
plt.ylabel('Histogram and density', fontsize = 17)
```

```
plt.show()
```






```
# Converting factor variables
# Selecting covariates
X = da.drop('SalePrice', axis = 1)
y = da['SalePrice']
y = np.log(y)

# Converting factors to dummies -----------------------------------
# Function of conversion
def dummy(var):
  X[var] = X[var].astype('category')
  X[var] = X[var].cat.codes
  return X

columns = X.columns

for index in range(0, len(columns)):
  if is_object_dtype(X[columns[index]]) == True:
    X = dummy(columns[index])
  else:
    X = X

X.dtypes # All ok, no factors

# Train and test split (automatic function)
```

```
## Id                   int64
```

```
## MSSubClass          int64
## MSZoning             int8
## LotFrontage       float64
## LotArea             int64
## Street               int8
## Alley                int8
## LotShape             int8
## LandContour          int8
## Utilities            int8
## LotConfig            int8
## LandSlope            int8
## Neighborhood         int8
## Condition1           int8
## Condition2           int8
## BldgType             int8
## HouseStyle           int8
## OverallQual         int64
## OverallCond         int64
## YearBuilt           int64
## YearRemodAdd        int64
## RoofStyle            int8
## RoofMatl             int8
## Exterior1st          int8
## Exterior2nd          int8
## MasVnrType           int8
## MasVnrArea        float64
## ExterQual            int8
## ExterCond            int8
## Foundation           int8
##                    ...
## HalfBath            int64
## BedroomAbvGr        int64
## KitchenAbvGr        int64
## KitchenQual          int8
## TotRmsAbvGrd        int64
## Functional           int8
## Fireplaces          int64
## FireplaceQu          int8
## GarageType           int8
## GarageYrBlt       float64
## GarageFinish         int8
## GarageCars          int64
## GarageArea          int64
## GarageQual           int8
## GarageCond           int8
## PavedDrive           int8
## WoodDeckSF          int64
## OpenPorchSF         int64
## EnclosedPorch       int64
## 3SsnPorch           int64
## ScreenPorch         int64
## PoolArea            int64
## PoolQC               int8
## Fence                int8
```

```
## MiscFeature         int8
## MiscVal            int64
## MoSold             int64
## YrSold             int64
## SaleType            int8
## SaleCondition       int8
## Length: 80, dtype: object
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 4)

# Fitting the model(s)
# ---------------------------
# 1. Linear regression
# 2. Random forests
# 3. SVM
# ---------------------------

# 1. Linear regression
model_lm = linear_model.LinearRegression()
model_lm.fit(X_train, y_train)

# Predictions and z-values
```

```
## LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
##          normalize=False)
```

```
y_pred_lm = model_lm.predict(X_test)
z_lm = (y_test - y_pred_lm)/np.std(y_test)

# 2. Random forests
model_rf = RandomForestRegressor()
model_rf.fit(X_train, y_train)

# Predictions and z-values
```

```
## RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
##            max_features='auto', 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=10, n_jobs=None,
##            oob_score=False, random_state=None, verbose=0, warm_start=False)
##
## /Users/brunawundervald/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/forest.py:248: FutureWa
##   "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
y_pred_rf = model_rf.predict(X_test)
z_rf = (y_test - y_pred_rf)/np.std(y_test)

# 3. SVM
model_svm = SVR()
model_svm.fit(X_train, y_train)

# Predictions and z-values
```

```
## SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
##   gamma='auto_deprecated', kernel='rbf', max_iter=-1, shrinking=True,
##   tol=0.001, verbose=False)
```

```
##
## /Users/brunawundervald/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:196: FutureWarning:
##   "avoid this warning.", FutureWarning)
```

```
y_pred_svm = model_svm.predict(X_test)
z_svm = (y_test - y_pred_svm)/np.std(y_test)

# Mean squared error, our prediction measure
print('LM:', round(mean_squared_error(y_test, y_pred_lm), 4))
```

```
## LM: 0.0173
```

```
print('RF:', round(mean_squared_error(y_test, y_pred_rf), 4))
```

```
## RF: 0.0198
```

```
print('SVM:', round(mean_squared_error(y_test, y_pred_svm), 4))
```

```
## SVM: 0.1463
```

```
# Quantile-quantile plots ---------------------------------------------------------
# If both sets of quantiles came from the same distribution, we should see the
# points forming a line that's roughly straight
plt.clf()
plt.figure(1)
```

```
plt.subplot(221)
```

```
stats.probplot(z_lm, dist="norm", plot=plt)
```

```
plt.title("Normal Q-Q plot - LM")
```

```
plt.subplot(222)
```

```
stats.probplot(z_rf, dist="norm", plot=plt)
```

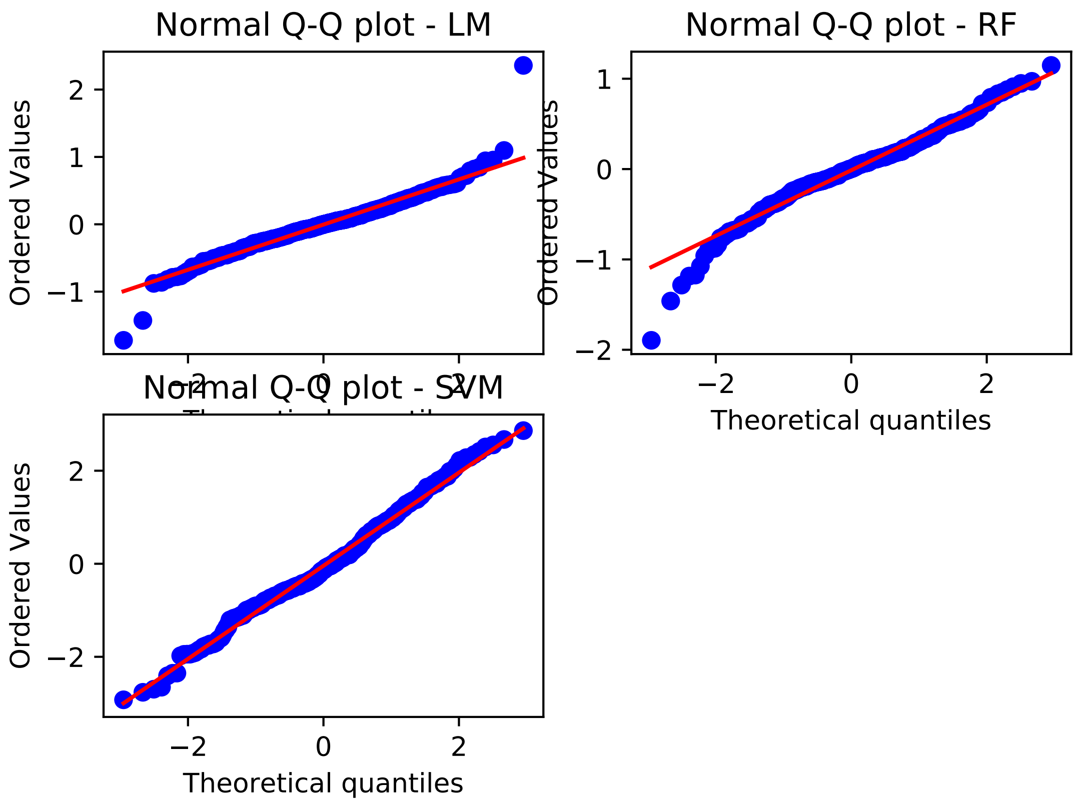
```
plt.title("Normal Q-Q plot - RF")
```

```
plt.subplot(223)
```

```
stats.probplot(z_svm, dist="norm", plot=plt)
```

```
plt.title("Normal Q-Q plot - SVM")
```

```
plt.show()
```



Conclusions

The linear regression model produced the smallest mean squared error, which indicates that it is predicting the response variable better than the other models. When looking at the Q-Q Plots, we can see that its residuals have the smallest magnitude, what shows that the predicted values are really closer to the reality. The plots also show that the linear regression model was not so influenced by the outliers as the Random Forest and the SVM.