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
```

```
# 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)
```

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

# Density plot of y in log scale -----------------------------------
plt.clf() # clean plot environment
plt.figure(figsize = (14, 10))
```

```
## <Figure size 1400x1000 with 0 Axes>
```

```
sns.distplot(y, bins = 30, kde = True, color = 'tomato', rug = True)
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x1a1de56c18>
##
```

```
## /Users/brunawundervald/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:1713: FutureWarning
##   return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

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

```
## Text(0.5,0,'Price Log')
```

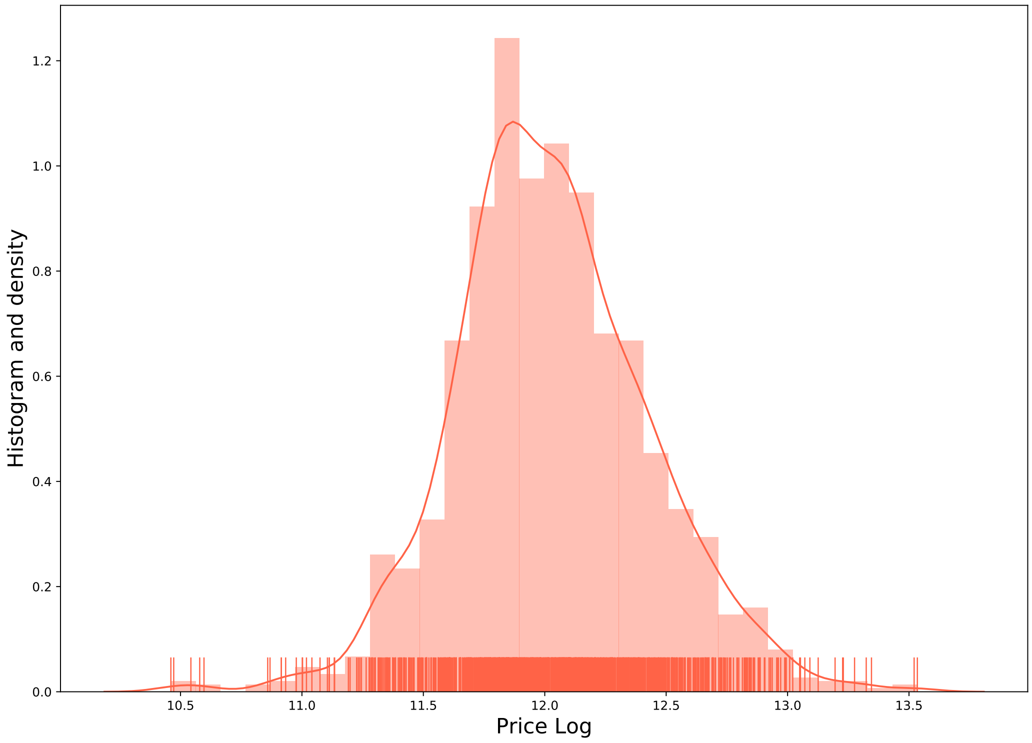
```
plt.ylabel('Histogram and density', fontsize = 17)
```

```
## Text(0,0.5,'Histogram and density')
```

```
plt.show()

# Converting factors to dummies ------------------------------------
```

```
## 'house-prices_files/figure-latex/unnamed-chunk-3-1.pdf'
```




```
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

# 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)


# Setting the model(s)
# ------------------------
# Linear regression
# Random forests
# SVM
# ------------------------
model_lm = linear_model.LinearRegression()
model_lm.fit(X_train, y_train)
```

```
## LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```
model_rf = RandomForestRegressor()
model_rf.fit(X_train, y_train)
```

```
## 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=1,
##            oob_score=False, random_state=None, verbose=0, warm_start=False)
```

```
model_svm = SVR()
model_svm.fit(X_train, y_train)

# Predictions
```

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

```
y_pred_lm = model_lm.predict(X_test)
y_pred_rf = model_rf.predict(X_test)
y_pred_svm = model_svm.predict(X_test)

# LM - Mean squared error:
mean_squared_error(y_test, y_pred_lm)

# RF - Mean squared error:
```

```
## 0.017330143259086012
```

```
mean_squared_error(y_test, y_pred_rf)

# SVM - Mean squared error:
```

```
## 0.017591142923888286
```

```
mean_squared_error(y_test, y_pred_svm)
```

```
## 0.14626955290505497
```