Raport - Adam Kusiakiewicz, Oskar Gierlicz

## Importowanie bibliotek oraz przedstawienie danych po ich załadowaniu

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy.stats import norm
from sklearn.preprocessing import StandardScaler
from scipy import stats
import warnings
from category_encoders import OrdinalEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, r2_score
warnings.filterwarnings('ignore')
%matplotlib inline
```

[2]:
```
# załadowanie danych
data = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
data
```

[2]:

| | Id | MSSubClass | MSZoning | LotFrontage | LotArea | Street | Alley | LotShape | LandContour | Utilities | ... | PoolArea | PoolQC | Fence | MiscFeature | MiscVal | MoSold | YrSold | S |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 60 | RL | 65.0 | 8450 | Pave | NaN | Reg | Lvl | AllPub | ... | 0 | NaN | NaN | NaN | 0 | 2 | 2008 | |
| 1 | 2 | 20 | RL | 80.0 | 9600 | Pave | NaN | Reg | Lvl | AllPub | ... | 0 | NaN | NaN | NaN | 0 | 5 | 2007 | |
| 2 | 3 | 60 | RL | 68.0 | 11250 | Pave | NaN | IR1 | Lvl | AllPub | ... | 0 | NaN | NaN | NaN | 0 | 9 | 2008 | |
| 3 | 4 | 70 | RL | 60.0 | 9550 | Pave | NaN | IR1 | Lvl | AllPub | ... | 0 | NaN | NaN | NaN | 0 | 2 | 2006 | |
| 4 | 5 | 60 | RL | 84.0 | 14260 | Pave | NaN | IR1 | Lvl | AllPub | ... | 0 | NaN | NaN | NaN | 0 | 12 | 2008 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 1455 | 1456 | 60 | RL | 62.0 | 7917 | Pave | NaN | Reg | Lvl | AllPub | ... | 0 | NaN | NaN | NaN | 0 | 8 | 2007 | |
| 1456 | 1457 | 20 | RL | 85.0 | 13175 | Pave | NaN | Reg | Lvl | AllPub | ... | 0 | NaN | MnPrv | NaN | 0 | 2 | 2010 | |
| 1457 | 1458 | 70 | RL | 66.0 | 9042 | Pave | NaN | Reg | Lvl | AllPub | ... | 0 | NaN | GdPrv | Shed | 2500 | 5 | 2010 | |
| 1458 | 1459 | 20 | RL | 68.0 | 9717 | Pave | NaN | Reg | Lvl | AllPub | ... | 0 | NaN | NaN | NaN | 0 | 4 | 2010 | |
| 1459 | 1460 | 20 | RL | 75.0 | 9937 | Pave | NaN | Reg | Lvl | AllPub | ... | 0 | NaN | NaN | NaN | 0 | 6 | 2008 | |

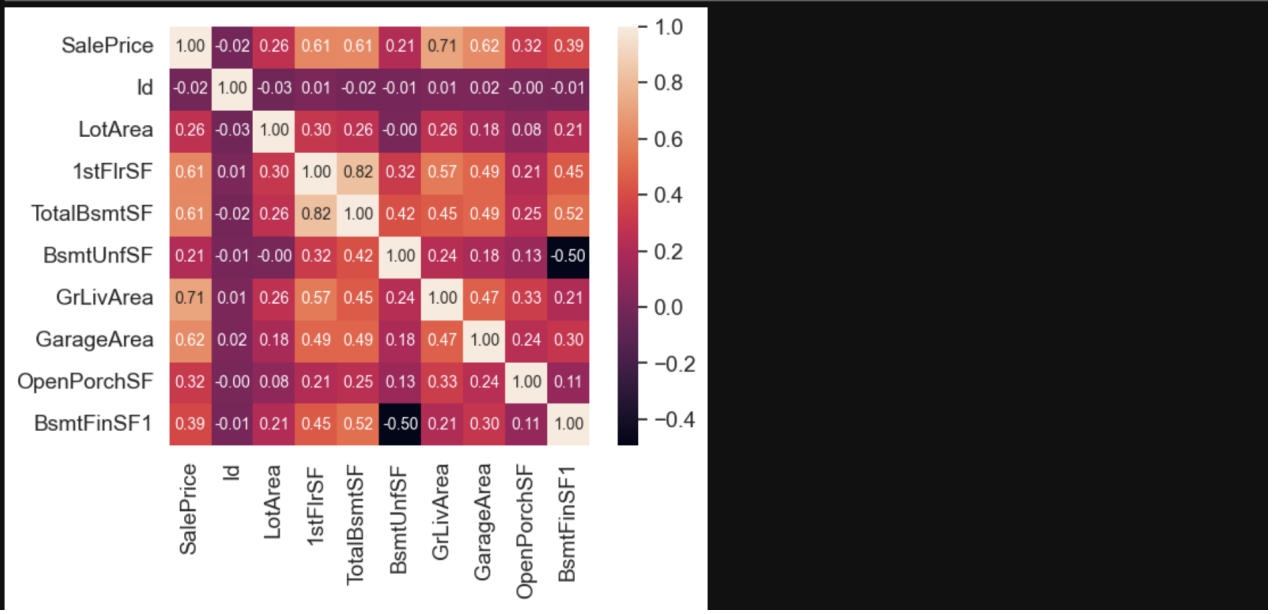
1460 rows × 81 columns

## Przedstawienie kolumn w train.csv oraz opis kluczowej dla nas kolumny SalePrice

```
[3]: # kolumny
     data.columns

[3]: Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
            'Alley', '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', '1stFlrSF', '2ndFlrSF',
            'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
            'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
            'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
            'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
            'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
            'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
            'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
            'SaleCondition', 'SalePrice'],
           dtype='object')

[4]: # opis SalePrice
     data['SalePrice'].describe()

[4]: count      1460.000000
     mean     180921.195890
     std       79442.502883
     min       34900.000000
     25%      129975.000000
     50%      163000.000000
     75%      214000.000000
     max      755000.000000
     Name: SalePrice, dtype: float64
```

## Dokładna mapa korelacji zawierająca 10 najbardziej istotnych korelacji

```
[6]: #dokładniejsza heatmapa, zawiera 10 najbardziej istotnych korelacji
     k = 10
     cols = corrmat.nlargest(k, 'SalePrice')['SalePrice'].index
     cm = np.corrcoef(data[cols].values.T)
     sns.set(font_scale=1.25)
     hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size': 10}, yticklabels=cols.values, xticklabels=cols.values)
     plt.show()
```




## Sprawdzenie pustych danych i usunięcie kolumn które je posiadają

[7]:
```
# sprawdzenie pustych danych
total = data.isnull().sum().sort_values(ascending=False)
percent = ((data.isnull().sum() / data.isnull().count()) * 100).sort_values(ascending=False).round(2)
empty_values = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
empty_values = empty_values.head(20)
empty_values
```

[7]:

| | Total | Percent |
|---|---|---|
| **PoolQC** | 1453 | 99.52 |
| **MiscFeature** | 1406 | 96.30 |
| **Alley** | 1369 | 93.77 |
| **Fence** | 1179 | 80.75 |
| **MasVnrType** | 872 | 59.73 |
| **FireplaceQu** | 690 | 47.26 |
| **LotFrontage** | 259 | 17.74 |
| **GarageYrBlt** | 81 | 5.55 |
| **GarageCond** | 81 | 5.55 |
| **GarageType** | 81 | 5.55 |
| **GarageFinish** | 81 | 5.55 |
| **GarageQual** | 81 | 5.55 |
| **BsmtFinType2** | 38 | 2.60 |
| **BsmtExposure** | 38 | 2.60 |
| **BsmtQual** | 37 | 2.53 |
| **BsmtCond** | 37 | 2.53 |
| **BsmtFinType1** | 37 | 2.53 |
| **MasVnrArea** | 8 | 0.55 |
| **Electrical** | 1 | 0.07 |
| **Id** | 0 | 0.00 |

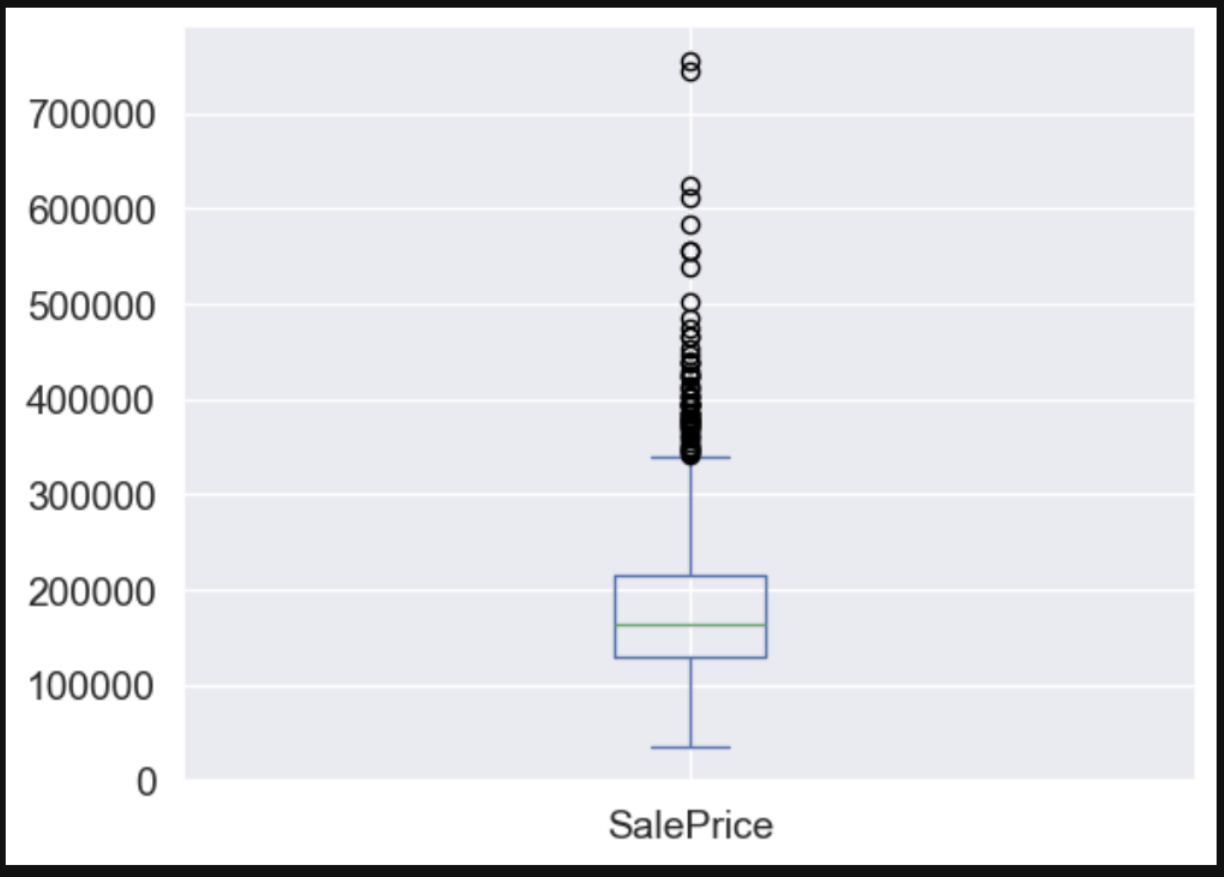
```
[8]: # usuwanie niepotrzebnych kolumn z pustymi danymi
     for i in range(len(empty_values) - 1):
         data = data.drop(empty_values['Total'].index[i], axis=1)

     data.columns
```

```
[8]: Index(['Id', 'MSSubClass', 'MSZoning', 'LotArea', 'Street', 'LotShape',
            'LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood',
            'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'OverallQual',
            'OverallCond', 'YearBuilt', 'YearRemodAdd', 'RoofStyle', 'RoofMatl',
            'Exterior1st', 'Exterior2nd', 'ExterQual', 'ExterCond', 'Foundation',
            'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
            'HeatingQC', 'CentralAir', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF',
            'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath',
            'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual', 'TotRmsAbvGrd',
            'Functional', 'Fireplaces', 'GarageCars', 'GarageArea', 'PavedDrive',
            'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
            'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
            'SaleCondition', 'SalePrice'],
           dtype='object')
```

Przedstawienie odstających wartości SalePrice

```
[9]: # odstające wartości SalePrice
     data['SalePrice'].plot.box()
```

```
[9]: <Axes: >
```




## Stworzenie danych do trenowania i ich optymalizacja

```
[11]: # usuwanie odstających wartości GrLivArea
data = data.drop(data[data['GrLivArea'] > 5000].index)
data['GrLivArea'].plot.box()# dane do uczenia, usunięcie kolumn typu Object uniemożliwiających nauczenie modelu XGBoosterem oraz
train_X = data.loc[:,:'SaleCondition']
train_y = data['SalePrice'].to_numpy()
test_X = test.copy()

numeric_cols = train_X.dtypes[train_X.dtypes != 'object'].index

train_X = train_X[numeric_cols]
test_X = test_X[numeric_cols]

train_X, test_X, train_y
```

## Utworzenie funkcji do sprawdzenia poprawności predykcji, nauka modelu

[12]:
```
#utworzenie funkcji do detekcji poprawności predykcji modelu
from sklearn import metrics
from sklearn.model_selection import cross_val_score

def cross_val(model):
    pred = cross_val_score(model, train_X, train_y, cv=10)
    return pred.mean()
```

[13]:
```
#przyuczenie modelu XGBRegressor danymi treningowymi
model = XGBRegressor()
model.fit(train_X, train_y, verbose=False)
```

[13]:
```
XGBRegressor

XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)
```

[14]:
```
predictions = model.predict(test_X)
```

[15]:
```
cross_val(model)
```

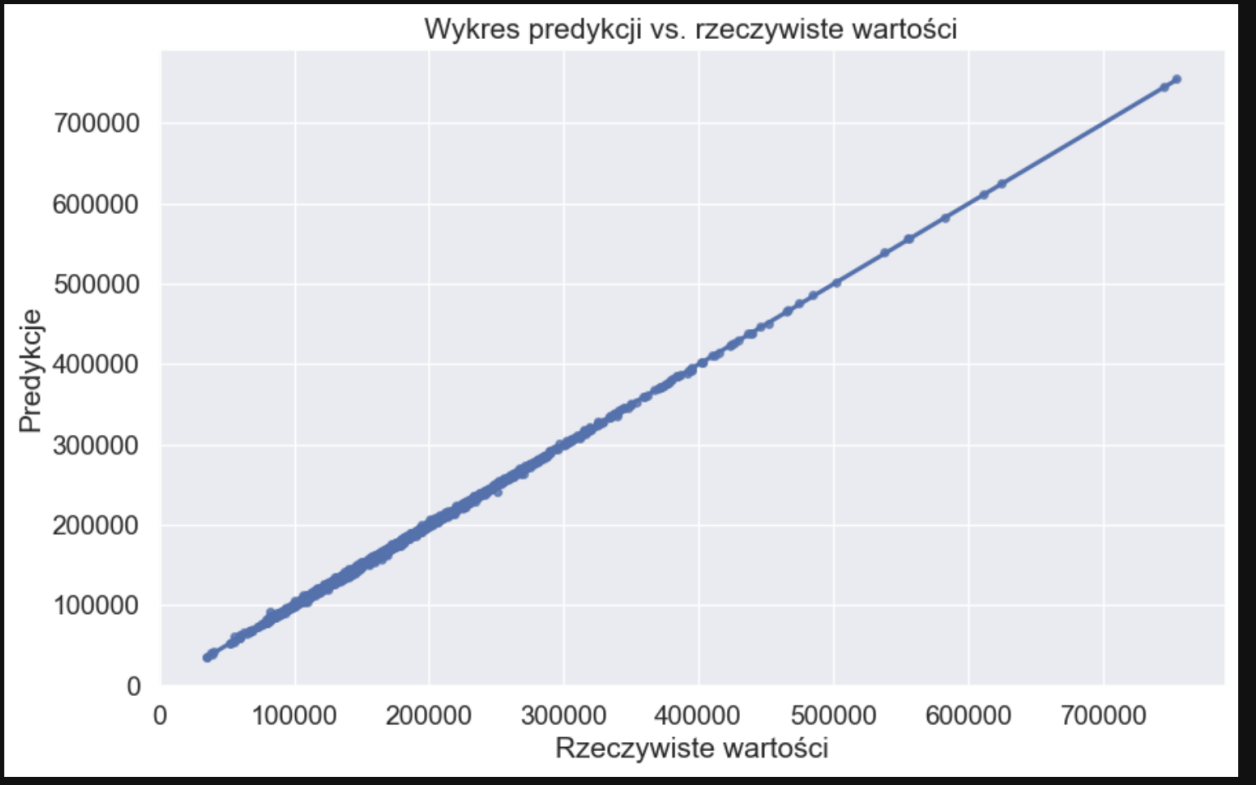
[15]:
```
0.8586327868564967
```

[16]:
```
# Ewaluacja modelu na danych treningowych
train_predictions = model.predict(train_X)
mse = mean_squared_error(train_y, train_predictions)
r2 = r2_score(train_y, train_predictions)

print(f'Mean Squared Error (MSE): {mse}')
print(f'R-squared (R2): {r2}')
```

```
Mean Squared Error (MSE): 2446343.272112987
R-squared (R2): 0.9996123567559827
```

Wykres przewidywanych wartości vs. rzeczywiste wartości

[17]:
```
# Wykres predykcji vs. rzeczywiste wartości
plt.figure(figsize=(10, 6))
sns.regplot(x=train_y, y=train_predictions, scatter_kws={'s': 15})
plt.xlabel('Rzeczywiste wartości')
plt.ylabel('Predykcje')
plt.title('Wykres predykcji vs. rzeczywiste wartości')
plt.show()
```




## Stworzenie submission

[19]:
```
#utworzenie DataFramu z wynikami
submission = pd.DataFrame({'Id' :test_X['Id'], 'SalePrice':predictions})
```

[20]:
```
submission.to_csv('submission.csv', index=False)
```

| | | | | | |
|---|---|---|---|---|---|
| 2958 | **oskargierlicz** | | 0.15300 | 5 | 1h |

Your Best Entry!
Your submission scored 0.15310, which is not an improvement of your previous score. Keep trying!