Machine Learning Final Project

Idrive Global

Autor: Adrian Iordache

Primul Pas… Incarcarea si Vizualizarea Datelor

In [11]:
```
data = pd.read_csv("train.csv")
data = data.drop('Id', axis = 1)
#display(data.head(n=20))

raw_features = data.drop("SalePrice", axis = 1)
labels = data["SalePrice"]
display(raw_features.head(n=10))
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	...	ScreenPorch	PoolArea	PoolQC	
0	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	0	NaN	
1	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	...	0	0	NaN	
2	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	...	0	0	NaN	
3	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	...	0	0	NaN	
4	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	...	0	0	NaN	
5	50	RL	85.0	14115	Pave	NaN	IR1	Lvl	AllPub	Inside	...	0	0	NaN	
6	20	RL	75.0	10084	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	0	NaN	
7	60	RL	NaN	10382	Pave	NaN	IR1	Lvl	AllPub	Corner	...	0	0	NaN	
8	50	RM	51.0	6120	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	0	NaN	
9	190	RL	50.0	7420	Pave	NaN	Reg	Lvl	AllPub	Corner	...	0	0	NaN	

10 rows × 79 columns

Se poate observa faptul ca datasetul contine valori NA (Non-Available), asa ca in primele faze este necesara o curatare a datelor.

Vom face asta prin eliminarea acelor coloane daca numarul de elemente NA din aceasta este peste 75% din numarul de total sau prin inlocuirea valorii NA cu valoarea cea mai frecventa din coloana respectiva.

In [12]:
```
raw_features = solve_nan_values(data = raw_features)
raw_features.head(n = 10)
```

Out[12]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	...	OpenPorchSF	EnclosedPor
0	60	RL	65.0	8450	Pave	Reg	Lvl	AllPub	Inside	Gtl	...	61	0
1	20	RL	80.0	9600	Pave	Reg	Lvl	AllPub	FR2	Gtl	...	0	0
2	60	RL	68.0	11250	Pave	IR1	Lvl	AllPub	Inside	Gtl	...	42	0
3	70	RL	60.0	9550	Pave	IR1	Lvl	AllPub	Corner	Gtl	...	35	272
4	60	RL	84.0	14260	Pave	IR1	Lvl	AllPub	FR2	Gtl	...	84	0
5	50	RL	85.0	14115	Pave	IR1	Lvl	AllPub	Inside	Gtl	...	30	0
6	20	RL	75.0	10084	Pave	Reg	Lvl	AllPub	Inside	Gtl	...	57	0
7	60	RL	60.0	10382	Pave	IR1	Lvl	AllPub	Corner	Gtl	...	204	228
8	50	RM	51.0	6120	Pave	Reg	Lvl	AllPub	Inside	Gtl	...	0	205
9	190	RL	50.0	7420	Pave	Reg	Lvl	AllPub	Corner	Gtl	...	4	0

10 rows × 75 columns

Dupa curatarea datelor, urmeaza pasul de convertire a datelor de tip non-numerical in date de tip numerical.

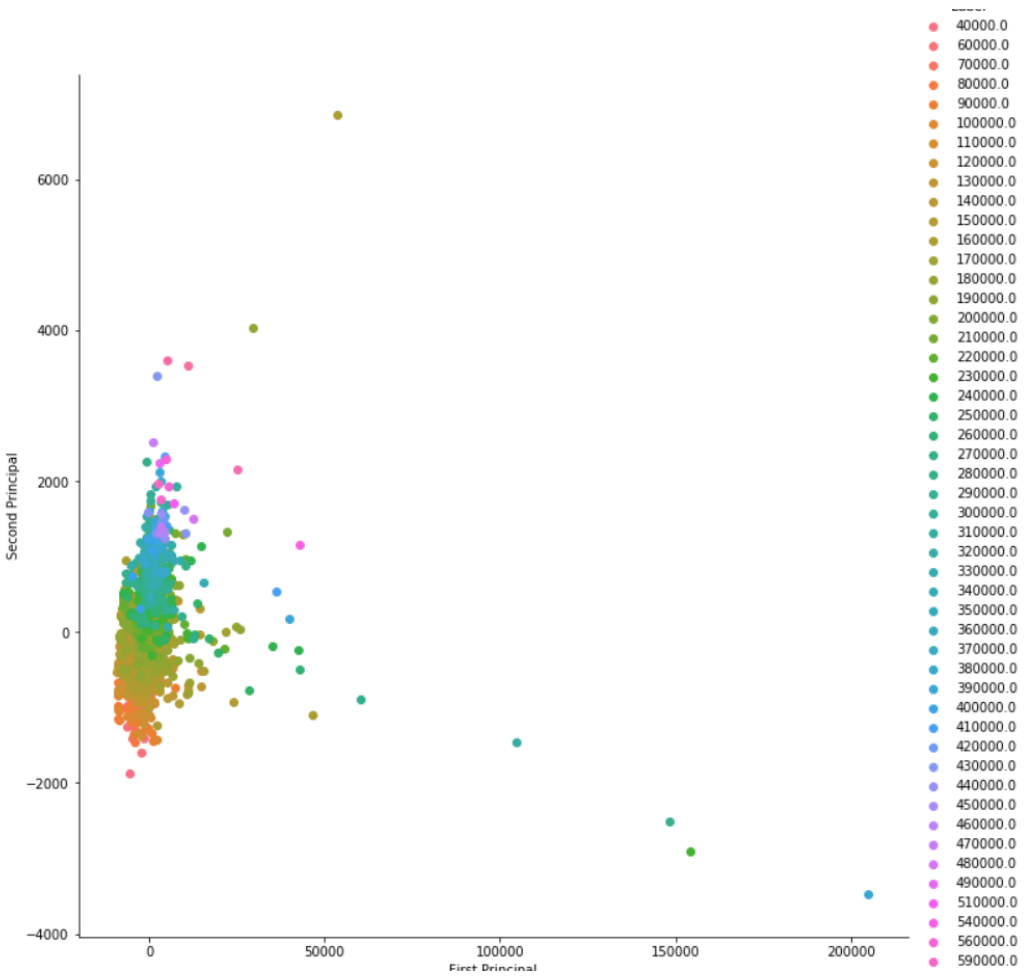
In [13]:
```
categorical_data = raw_features.select_dtypes(include = ['object']).copy()
categorical_data.head(n = 10)

categorical_data = convert_to_numerical(data = categorical_data)
categorical_data.head(n = 10)
```

Out[13]:

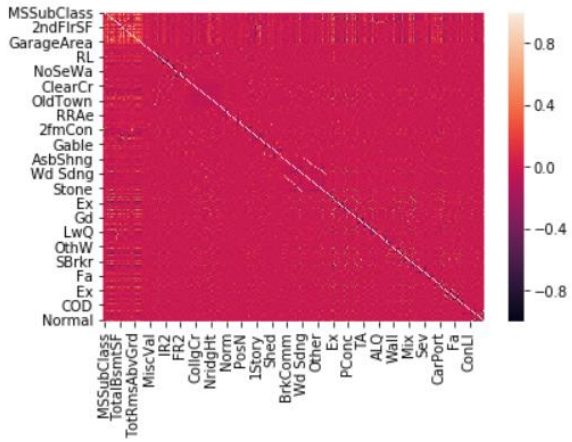
	C (all)	FV	RH	RL	RM	Grvl	Pave	IR1	IR2	IR3	...	ConLw	New	Oth	WD	Abnorml	AdjLand	Alloca	Family	Normal	Partial
0	0	0	0	1	0	0	1	0	0	0	...	0	0	0	1	0	0	0	0	1	0
1	0	0	0	1	0	0	1	0	0	0	...	0	0	0	1	0	0	0	0	1	0
2	0	0	0	1	0	0	1	1	0	0	...	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	0	0	1	1	0	0	...	0	0	0	1	1	0	0	0	0	0
4	0	0	0	1	0	0	1	1	0	0	...	0	0	0	1	0	0	0	0	1	0
5	0	0	0	1	0	0	1	1	0	0	...	0	0	0	1	0	0	0	0	1	0
6	0	0	0	1	0	0	1	0	0	0	...	0	0	0	1	0	0	0	0	1	0
7	0	0	0	1	0	0	1	1	0	0	...	0	0	0	1	0	0	0	0	1	0
8	0	0	0	0	1	0	1	0	0	0	...	0	0	0	1	1	0	0	0	0	0
9	0	0	0	1	0	0	1	0	0	0	...	0	0	0	1	0	0	0	0	1	0

10 rows × 239 columns

Vizualizarea datelor obtiunute pe un grafic ce reduce dimensionalitatea intr-un cadru 2D prin metoda PCA (Principal Component Analysis).

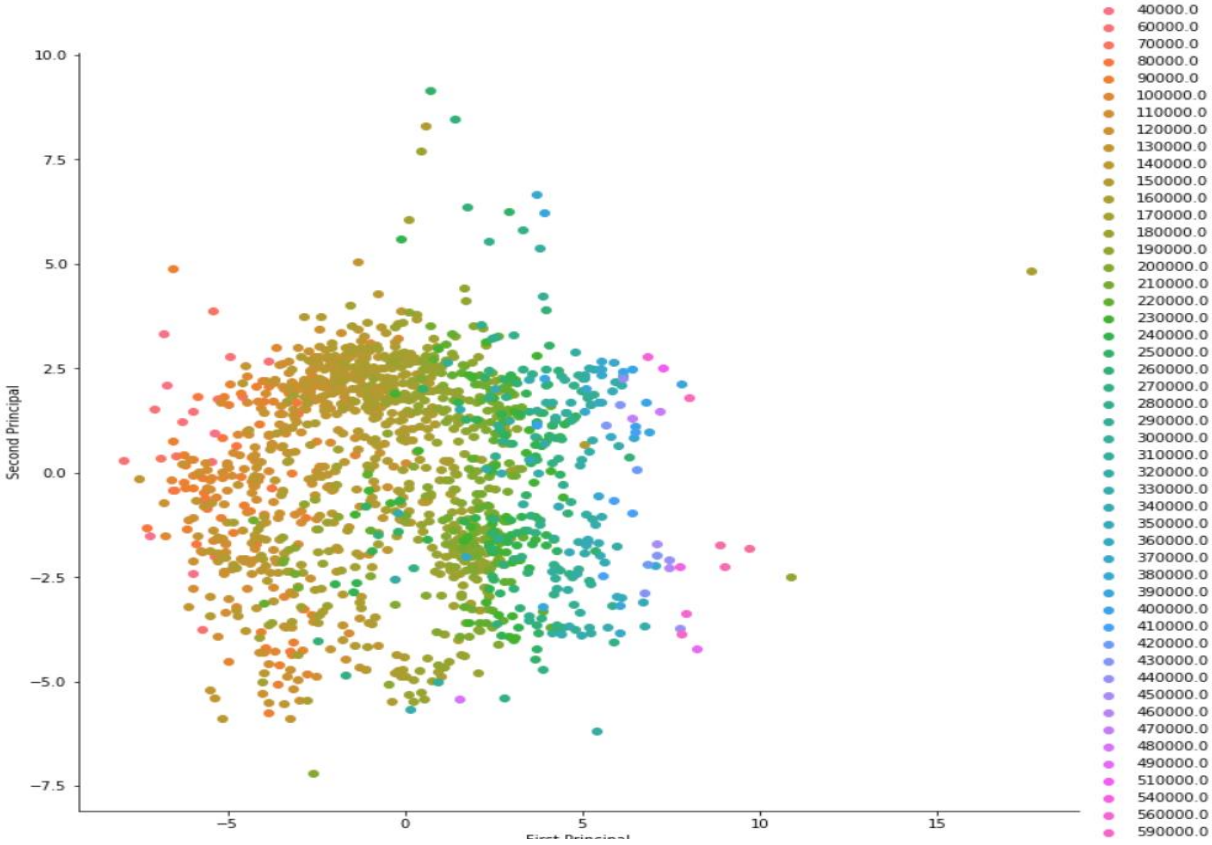


Observandu-se numarul mare de dimensiuni al datasetului obtinut vom incerca prin nivelul de corelatie intre coloane sa vedem daca putem renunta la un numar de dimensiuni pentru o mai buna eroare a modelului.



Dupa incercare mai multor metode de preprocesare precum Batch Normalization, MinMaxScaler, MaxAbsScaler, sklearn.preprocessing, Binarizer, Normalizer, StandardScaler.

S-a ajuns la concluzia ca cea mai eficienta metoda de preprocesare este StandardScaler cu eliminarea unor dimensiuni ce dispun de un indice de corelatie mai mare de 90%.



Dupa acesti pasi de preprocesare a datelor puntem incepe antrenarea unor modele in cautare unei erori mai mica de 23%.

Primul model antrenat pentru o eroare initiala estimativa a fost un RandomForestRegressor neparametrizat ce a obtinut o eroare de 38%.

Pe parcursul sesiunilor de antrenamente cu GridSearchCV si CrossValidation pentru evitarea posibilei situatii de overfit s-au folosit urmatorii algoritmi de Machine Learning cautandu-se minimizarea functiei de eroare RMSLE (Root Mean Squared Logarithmic Error).

1. KNN Regressor (K-Nearest Neighbors)

```
regressor = KNeighborsRegressor(n_neighbors = 13, weights = 'distance')
regressor.fit(X_train,y_train)
y_predict = regressor.predict(X_test)
error = RMSLE(y_test, y_predict)
print ("KNeighborsRegressor Error:", error)

#Training Score:  0.6751081074062033
#Best score: -0.2035467062998057
#Best parameters: {'n_neighbors': 13, 'weights': 'distance'}
#KNeighborsRegressor Error: 0.20294638416160402
```

```
Training set has 1095 samples.
Testing set has 365 samples.
KNeighborsRegressor Error: 0.20294638416160402
```

2. Support Vector Machine

```
regressor = svm.SVR(C = 0.001, gamma = 1, kernel = 'poly')
regressor.fit(X_train,y_train)
y_predict = regressor.predict(X_test)
error = RMSLE(y_test, y_predict)
print ("Support Vector Machine Error:", error)

#Training Score:  -0.049315027243430176
#Best score: -0.3414677011904274
#Best parameters: {'C': 0.001, 'gamma': 1, 'kernel': 'poly'}
#Support Vector Machine Error: 0.3427940440637342
```

```
Training set has 1095 samples.
Testing set has 365 samples.
Support Vector Machine Error: 0.3427940440637342
```

3. MLPRegressor (Multilayer-Perceptron)

```
regressor = MLPRegressor(activation = 'logistic', solver = 'lbfgs', learning_rate = 'adaptive', max_iter = 1000, alpha
= 0.001, hidden_layer_sizes = 700, random_state = 42)
regressor.fit(X_train,y_train)
y_predict = regressor.predict(X_test)
error = RMSLE(y_test, y_predict)
print ("MLPRegressor Error:", error)

#Training Score:  -0.18151158315872673
#Best score: -0.1814913343616067
#Best parameters: {'activation': 'logistic', 'alpha': 0.001, 'hidden_layer_sizes': 700, 'learning_rate': 'adaptive', 'm
ax_iter': 1000, 'solver': 'lbfgs'}
#MLPRegressor Error: 0.16807183194855363
```

```
Training set has 1095 samples.
Testing set has 365 samples.
MLPRegressor Error: 0.16807183194855363
```

Fiecare dintre acestia a obtinut o eroare decenta, insa algoritmul care a adus rezultatul final a fost un AdaBoostRegressor parametrizat ce foloseste ca estimator un RandomForestRegressor cu parametri gasiti prin hiper tunare cu GridSearchCV.

```
regressor = RandomForestRegressor(n_estimators = 25, max_depth = 750)
ada = AdaBoostRegressor(n_estimators = 20, base_estimator = regressor, learning_rate = 0.5)
ada.fit(X_train,y_train)
y_predict = ada.predict(X_test)
error = RMSLE(y_test, y_predict)
print ("RandomForestRegressor Error:",error)
#RandomForestRegressor Error: 0.13995298236641165


#Training Score:  -0.15247668929852423
#Best score: -0.1523665243952441
#Best parameters: {'max_depth': 25, 'n_estimators': 750}
#RandomForestRegressor Error: 0.14002055610418848
```

```
Training set has 1095 samples.
Testing set has 365 samples.
RandomForestRegressor Error: 0.14087259513840988
```

Limita de eroare ce trebuia depasita era de 23%, iar modelul antrenat ajungand la o eroare de 14% denota finalizarea cu success a acestui proiect.