IFT870/BIN710 - Corrigé - Exercice : Prétraitement et Représentation des données

Dans la suite de cet exercice, nous procéderons à l'analyse du jeu de données pokedex.csv. Ce fichier de la documentation en trois (3) variétés :

- https://www.kaggle.com/datasets/mariosios/pokemon-dataset-extended-version

Question 1 : Jeu de données et description statistique des attributs

- Charger le jeu de données et effectuer une description statistique des attributs.

Corrigé Question 1

```
# Importation des bibliothèques
import numpy as np
import pandas as pd
import matplotlib as mp
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
# Lecture du jeu de données et affichage des descriptions statistiques des attributs.
pokedex = pd.read_csv("./pokedex.csv")
print("\n\n\n
```

Commentaires:

- Vérifier que les types des attributs correspondent bien aux valeurs.
- Vérifier que les attributs corrélés ont des valeurs cohérentes.

In [32]:
```
valeurs = [data[att].values[5] for att in data.columns]
pd.DataFrame({'Types': data.dtypes, '5 1eres valeurs': valeurs}).transpose()
```

Out[32]:

	PID	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Lot Shape	Land Contour	Utilities	Lot Config	...	Pool Area	Pool QC	Misc Feature	Misc Val	Mo Sold	Yr Sold	Sale Type	Sale Condition	SalePrice	HasFence
Types	int64	int64	object	float64	int64	object	object	object	object	object	...	int64	object	object	int64	int64	int64	object	object	int64	int64
5 1eres valeurs	[526301100, 526301375, 526351010, 526353030, 5...]	[20, 20, 20, 20, 60]	[RL, RH, RL, RL, RL]	[141.0, 80.0, 93.0, 93.0, 74.0]	[31770, 11622, 14267, 11160, 13830]	[Pave, Pave, Pave, Pave, Pave]	[IR1, Reg, IR1, Reg, IR1]	[Lvl, Lvl, Lvl, Lvl, Lvl]	[AllPub, AllPub, AllPub, AllPub, AllPub]	[Corner, Inside, Corner, Corner, Inside]	...	[0, 0, 0, 0, 0]	[nan, nan, nan, nan, nan]	[nan, nan, Gar2, 12500, nan]	[0, 0, 0, 0, 0]	[5, 6, 6, 4, 3]	[2010, 2010, 2010, 2010, 2010]	[WD , WD , WD , WD , WD]	[Normal, Normal, Normal, Normal, Normal]	[215000, 105000, 172000, 244000, 189900]	[0, 1, 0, 0, 1]

2 rows × 78 columns

Question 4 : Transformation et représentation des données

- A la suite de votre analyse, pouvez-vous effectuer une sélection des attributs pertinents ? Si oui, quels sont les attributs que vous choisissez et pourquoi ?
- Serait-il judicieux d'appliquer une décomposition en composantes principales (PCA) à votre jeu de données ? Si oui, combien de composantes pensez-vous qu'il serait raisonnable de conserver ?.

Corrigé Question 4

- A la suite de votre analyse, pouvez-vous effectuer une sélection des attributs pertinents ? Si oui, quels sont les attributs que vous choisissez et pourquoi ?

Commentaires:

Oui, il est possible de sélectionner certains attributs en raison de la redondance d'informations et de la présence d'attributs qui ne sont pas nécessairement utiles, ce qui peut être analysé par une étude de la corrélation entre les attributs.

In [33]:
```
data.select_dtypes(include=['float64', 'int64', 'int32'])
```

Out[33]:

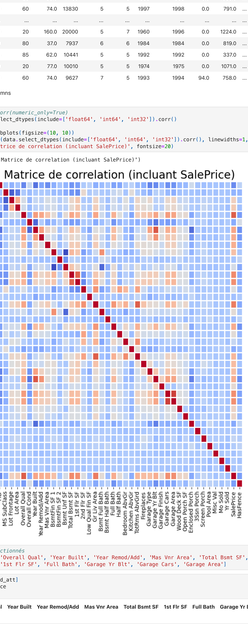
	PID	MS SubClass	Lot Frontage	Lot Area	Overall Qual	Overall Cond	Year Built	Year Remod/Add	Mas Vnr Area	BsmtFin SF 1	...	Open Porch SF	Enclosed Porch	3Ssn Porch	Screen Porch	Pool Area	Misc Val	Mo Sold	Yr Sold	SalePrice	HasFence
Order																					
1	526301100	20	141.0	31770	6	5	1960	1960	112.0	639.0	...	62	0	0	0	0	0	5	2010	215000	0
2	526350040	20	80.0	11622	5	6	1961	1961	0.0	468.0	...	0	0	0	120	0	0	6	2010	105000	1
3	526351010	20	81.0	14267	6	6	1958	1958	108.0	923.0	...	36	0	0	0	0	12500	6	2010	172000	0
4	526353030	20	93.0	11160	7	5	1968	1968	0.0	1065.0	...	0	0	0	0	0	0	4	2010	244000	0
5	527050010	60	74.0	13830	5	5	1997	1998	0.0	791.0	...	34	0	0	0	0	0	3	2010	189900	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
2925	923275080	20	160.0	20000	5	7	1960	1996	0.0	1224.0	...	0	0	0	0	0	0	9	2006	131000	0
2926	923275080	80	37.0	7937	6	6	1984	1984	0.0	819.0	...	0	0	0	0	0	0	3	2006	142500	1
2928	923400125	85	62.0	10441	5	5	1992	1992	0.0	337.0	...	32	0	0	0	0	700	7	2006	132000	1
2929	924100070	20	77.0	10010	5	5	1974	1975	0.0	1071.0	...	38	0	0	0	0	0	4	2006	170000	0
2930	924151050	60	74.0	9627	7	5	1993	1994	94.0	758.0	...	48	0	0	0	0	0	11	2006	188000	0

2432 rows × 41 columns

In [34]:
```
#correl = data.corr(numeric_only=True)
correl = data.select_dtypes(include=['float64', 'int64', 'int32']).corr()

fig, ax = plt.subplots(figsize=(10, 10))
ax = sns.heatmap(data.select_dtypes(include=['float64', 'int64', 'int32']).corr(), linewidths=1, cmap="coolwarm", annot_kws={"size": 10})
ax.set_title('Matrice de correlation (incluant SalePrice)', fontsize=20)
```

Out[34]: Text(0.5, 1.0, 'Matrice de correlation (incluant SalePrice)')



In [35]:
```
# Attributs sélectionnés
Selected_att = ['Overall Qual', 'Year Built', 'Year Remod/Add', 'Mas Vnr Area', 'Total Bsmt SF',
                '1st Flr SF', 'Full Bath', 'Garage Yr Blt', 'Garage Cars', 'Garage Area']
```

In [36]:
```
X = data[Selected_att]
y = data.SalePrice
X
```

Out[36]:

	Overall Qual	Year Built	Year Remod/Add	Mas Vnr Area	Total Bsmt SF	1st Flr SF	Full Bath	Garage Yr Blt	Garage Cars	Garage Area
Order										
1	6	1960	1960	112.0	1080.0	1656	1	1960.0	2.0	528.0
2	5	1961	1961	0.0	882.0	896	1	1961.0	1.0	730.0
3	6	1958	1958	108.0	1329.0	1329	1	1958.0	1.0	312.0
4	7	1968	1968	0.0	2110.0	2110	2	1968.0	2.0	522.0
5	5	1997	1998	0.0	928.0	928	2	1997.0	2.0	482.0
---	---	---	---	---	---	---	---	---	---	---
2925	5	1960	1996	0.0	1224.0	1224	1	1960.0	2.0	576.0
2926	6	1984	1984	0.0	1003.0	1003	1	1984.0	2.0	588.0
2928	5	1992	1992	0.0	912.0	970	1	NaN	0.0	0.0
2929	5	1974	1975	0.0	1389.0	1389	1	1975.0	2.0	418.0
2930	7	1993	1994	94.0	996.0	996	2	1993.0	3.0	650.0

2432 rows × 10 columns

In [37]:
```
print("Valeurs manquantes avant imputation:\n", X.isnull().sum())
```

```
Valeurs manquantes avant imputation:
 Overall Qual        0
Year Built          0
Year Remod/Add      0
Mas Vnr Area       17
Total Bsmt SF       1
1st Flr SF          0
Full Bath           0
Garage Yr Blt     148
Garage Cars         1
Garage Area         1
dtype: int64
```

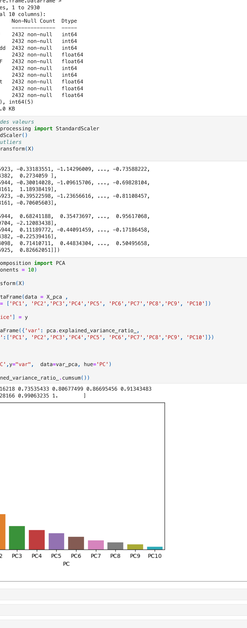
In [38]:
```
# Imputation
for att in Selected_att:
    X.loc[:, att] = X.loc[:,att].fillna(X[att].mode()[0])
    print("Valeurs manquantes après imputation:\n", X.isnull().sum())
```

```
Valeurs manquantes après imputation:
 Overall Qual        0
Year Built          0
Year Remod/Add      0
Mas Vnr Area        0
Total Bsmt SF       0
1st Flr SF          0
Full Bath           0
Garage Yr Blt       0
Garage Cars         0
Garage Area         0
dtype: int64
```

In [39]:
```
correl = X.select_dtypes(include=['float64', 'int64', 'int32']).corr()

fig, ax = plt.subplots(figsize=(10, 10))
ax = sns.heatmap(X.select_dtypes(include=['float64', 'int64', 'int32']).corr(), linewidths=1, cmap="coolwarm", annot_kws={"size": 10})
ax.set_title('Matrice de correlation (avec attributs selectionnes incluant SalePrice)', fontsize=20)
```

Out[39]: Text(0.5, 1.0, 'Matrice de correlation (avec attributs selectionnes incluant SalePrice)')



- Serait-il judicieux d'appliquer une décomposition en composantes principales (PCA) à votre jeu de données ? Si oui, combien de composantes pensez-vous qu'il serait raisonnable de conserver ?.

Commentaires:

Oui, une décomposition en composantes principales permettrait de renforcer notre sélection d'attributs en analysant la variance entre les attributs après la transformation.

In [ ]:

In [40]:
```
# Y'a t-il des attributs catégoriels? (Réponse: Non, il n'y a pas d'attributs catégoriels. Si tel avait été le cas, une transformation préalable aurait probablement été nécessai
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2432 entries, 1 to 2930
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Overall Qual    2432 non-null   int64
 1   Year Built      2432 non-null   int64
 2   Year Remod/Add  2432 non-null   int64
 3   Mas Vnr Area    2432 non-null   float64
 4   Total Bsmt SF   2432 non-null   float64
 5   1st Flr SF      2432 non-null   int64
 6   Full Bath       2432 non-null   int64
 7   Garage Yr Blt   2432 non-null   float64
 8   Garage Cars     2432 non-null   float64
 9   Garage Area     2432 non-null   float64
dtypes: float64(5), int64(5)
memory usage: 209.0 KB
```

In [41]:
```
# Normalisation des valeurs
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
# Sensible aux outliers
X = scaler.fit_transform(X)
X
```

Out[41]:
```
array([[-0.06195923, -0.33183551, -1.54296089,  ..., -0.73588222,
        -0.32614382, -0.27348959 ],
       [-0.75635944, -0.30014820, -1.09615706,  ..., -0.69828184,
        -0.95318161, -1.10938439],
       [-0.06195923, -0.39522598, -1.23656616,  ..., -0.81180457,
        -0.95318161, -0.70605681],
       ...,
       [-0.75635944,  0.68241188,  0.35473697,  ...,  0.95617068,
         -2.23250704, -2.12083438],
       [-0.75635944,  0.11189772,  -0.44091459,  ..., -0.17186458,
         -0.32614382, -0.22539416],
       [ 0.63524089,  0.67418711,  0.44834304,  ...,  0.50495658,
          1.60546925,  0.82662051]])
```

In [42]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components = 10)
pca.fit(X)
X_pca = pca.transform(X)

data_pca = pd.DataFrame(data = X_pca',
                        columns = ['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10'])

data_pca['SalePrice'] = y

var_pca = pd.DataFrame({'var': pca.explained_variance_ratio_,
             'PC': ['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10']})

var_pca

sns.barplot(x='PC',y="var", data=var_pca, hue='PC')

print(pca.explained_variance_ratio_.cumsum())
```

```
[0.51787  0.64916218 0.75535433 0.80677499 0.86695456 0.91343483
 0.94678736 0.97221866 0.99003235 1.        ]
```


In [ ]:

In [ ]:

In [ ]:

In [ ]: