L'objectif de ce travail est de déterminer les composantes principales qui influent le plus sur le prix de vente d'un appartement, dans la ville de Tehran (Iran), à l'aide du modèle de Machine Learning non supervisé ACP (Analyse en Composantes Principales).

Les données ont été récupérées depuis l'adresse suivante:

https://archive.ics.uci.edu/ml/datasets/Residential+Building+Data+Set

Ce dataset (jeu de données) contient les caractéristiques des ensembles immobiliers déjà construits dans Tehran, notamment le coût de construction, le prix de vente, quelques indicateurs économiques de base tel l'indice des prix à la consommation ainsi que des variables propres au projet.

Vous trouverez le détail des différentes variables utilisées dans le fichier README.

In [1]:
```
# Importation des premiers packages nécessaires

%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Chargement du jeu de données, au format xlsx, dans un dataframe nommé df:

df = pd.read_excel('Residential-Building-Data-Set.xlsx', header = None)
```

In [3]:
```
# Dimensions de notre dataframe:

df.shape
```

Out[3]: (374, 109)

In [4]:
```
# Changement de quelques paramètres afin d'afficher toutes les colonnes, au nombre de 109:

pd.set_option('display.max_columns', 109)
```

In [5]:
```
# Aperçu des 5 premières lignes de notre dataframe:

df.head()
```

Out[5]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	PROJECT DATES (PERSIAN CALENDAR)	NaN	NaN	NaN	PROJECT PHYSICAL AND FINANCIAL VARIABLES	NaN	NaN	NaN	NaN	NaN	NaN	NaN	ECONOMIC VARIABLES AND INDICES IN TIME LAG 1	NaN	NaN	NaN	NaN
1	START YEAR	START QUARTER	COMPLETION YEAR	COMPLETION QUARTER	V-1	V-2	V-3	V-4	V-5	V-6	V-7	V-8	V-11	V-12	V-13	V-14	V-15
2	81	1	85	1	1	3150	920	598.5	190	1010.84	16	1200	6713	56.2	61.52	6.11	320957
3	84	1	89	4	1	7600	1140	3040	400	963.81	23	2900	3152	106	103.03	3.15	685698
4	78	1	81	4	1	4800	840	480	100	689.84	15	630	1627	41	41.25	1.74	160402

In [6]:
```
# Suppression de la première ligne de notre dataframe (index = 0):

df.drop(df.index[0])
```

Out[6]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	START YEAR	START QUARTER	COMPLETION YEAR	COMPLETION QUARTER	V-1	V-2	V-3	V-4	V-5	V-6	V-7	V-8	V-11	V-12	V-13	V-14	V-15	V-16	V-17
2	81	1	85	1	1	3150	920	598.5	190	1010.84	16	1200	6713	56.2	61.52	6.11	320957	3485.8	64.5
3	84	1	89	4	1	7600	1140	3040	400	963.81	23	2900	3152	106	103.03	3.15	685698	3526.1	105.5
4	78	1	81	4	1	4800	840	480	100	689.84	15	630	1627	41	41.25	1.74	160402	1217.5	34.4
5	72	2	73	2	1	685	202	13.7	20	459.54	4	140	2580.93	12.1	10.03	1.24	38193.6	287.2	13.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
369	83	4	86	1	20	1350	350	108	80	251.37	9	830	2700	103	101.84	2.65	625829	4386.9	100.4
370	81	2	82	4	20	600	150	36	60	299.55	6	570	6713	59.1	65.61	5.65	339326	3878.4	70.2
371	84	4	86	3	20	1900	430	285	150	364.41	7	640	2918	120.2	107.2	2.29	832124	5136.1	109.0
372	82	3	84	4	20	510	160	30.6	60	245.28	9	790	2247	76.9	79.93	2.04	463829	4107.4	88.2
373	80	1	81	3	20	890	230	35.6	40	237.03	6	350	7196	51.3	56.13	5.97	249111	2562.3	52.8

In [7]:
```
# Récupération de la liste des noms de la première colonne:

names = list(df.iloc[1])
```

In [8]:
```
print(names)
```

```
['START YEAR', 'START QUARTER', 'COMPLETION YEAR', 'COMPLETION QUARTER', 'V-1', 'V-2', 'V-3', 'V-4', 'V-5', 'V-6', 'V
-7', 'V-8', 'V-11', 'V-12', 'V-13', 'V-14', 'V-15', 'V-16', 'V-17', 'V-18', 'V-19', 'V-20', 'V-21', 'V-22', 'V-23', '
V-24', 'V-25', 'V-26', 'V-27', 'V-28', 'V-29', 'V-11', 'V-12', 'V-13', 'V-14', 'V-15', 'V-16', 'V-17', 'V-18', 'V-19
', 'V-20', 'V-21', 'V-22', 'V-23', 'V-24', 'V-25', 'V-26', 'V-27', 'V-28', 'V-29', 'V-11', 'V-12', 'V-13', 'V-14', 'V
-15', 'V-16', 'V-17', 'V-18', 'V-19', 'V-20', 'V-21', 'V-22', 'V-23', 'V-24', 'V-25', 'V-26', 'V-27', 'V-28', 'V-29',
'V-11', 'V-12', 'V-13', 'V-14', 'V-15', 'V-16', 'V-17', 'V-18', 'V-19', 'V-20', 'V-21', 'V-22', 'V-23', 'V-24', 'V-25
', 'V-26', 'V-27', 'V-28', 'V-29', 'V-11', 'V-12', 'V-13', 'V-14', 'V-15', 'V-16', 'V-17', 'V-18', 'V-19', 'V-20', 'V
-21', 'V-22', 'V-23', 'V-24', 'V-25', 'V-26', 'V-27', 'V-28', 'V-29', 'V-9', 'V-10']
```

In [9]:
```
# Mise de la liste "names" comme nouvelle appélation des colonnes:

df.set_axis(names, axis = 'columns', inplace = True)
```

In [10]:
```
df.head()
```

Out[10]:

	START YEAR	START QUARTER	COMPLETION YEAR	COMPLETION QUARTER	V-1	V-2	V-3	V-4	V-5	V-6	V-7	V-8	V-11	V-12	V-13	V-14	V-
0	PROJECT DATES (PERSIAN CALENDAR)	NaN	NaN	NaN	PROJECT PHYSICAL AND FINANCIAL VARIABLES	NaN	NaN	NaN	NaN	NaN	NaN	NaN	ECONOMIC VARIABLES AND INDICES IN TIME LAG 1	NaN	NaN	NaN	Na
1	START YEAR	START QUARTER	COMPLETION YEAR	COMPLETION QUARTER	V-1	V-2	V-3	V-4	V-5	V-6	V-7	V-8	V-11	V-12	V-13	V-14	V-
2	81	1	85	1	1	3150	920	598.5	190	1010.84	16	1200	6713	56.2	61.52	6.11	3209
3	84	1	89	4	1	7600	1140	3040	400	963.81	23	2900	3152	106	103.03	3.15	6856
4	78	1	81	4	1	4800	840	480	100	689.84	15	630	1627	41	41.25	1.74	1604

In [11]:
```
# Suppression définitive des deux première lignes:

df.drop(df.index[0:2], inplace = True)
```

In [12]:
```
df.head()
```

Out[12]:

	START YEAR	START QUARTER	COMPLETION YEAR	COMPLETION QUARTER	V-1	V-2	V-3	V-4	V-5	V-6	V-7	V-8	V-11	V-12	V-13	V-14	V-15	V-16	V
2	81	1	85	1	1	3150	920	598.5	190	1010.84	16	1200	6713	56.2	61.52	6.11	320957	3485.8	6
3	84	1	89	4	1	7600	1140	3040	400	963.81	23	2900	3152	106	103.03	3.15	685698	3526.1	10
4	78	1	81	4	1	4800	840	480	100	689.84	15	630	1627	41	41.25	1.74	160402	1217.5	3
5	72	2	73	2	1	685	202	13.7	20	459.54	4	140	2580.93	12.1	10.03	1.24	38193.6	287.2	1
6	87	1	90	2	1	3000	800	1230	410	631.91	13	5000	6790	203.8	162.84	6.46	1640293	10855.3	22

Nous commençons à avoir un dataframe qui prend forme pour la future analyse, à l'aide d'un algorithme de Machine Learning que nous allons déterminer plus tard.

In [13]:
```
df.shape
```

Out[13]: (372, 109)

In [14]:
```
# Quelques statistiques de base:

df.describe()
```

Out[14]:

	START YEAR	START QUARTER	COMPLETION YEAR	COMPLETION QUARTER	V-1	V-2	V-3	V-4	V-5	V-6	V-7	V-8	V-11	V-12	V-13	V-14	V-15	
count	372	372	372	372	372	372.0	372.0	372.0	372	372.00	372	372	372.0	372.0	372.00	372.00	372.0	
unique	17	4	18	4	20	216.0	105.0	337.0	50	314.00	16	121	66.0	67.0	67.00	65.00	67.0	
top	86	1	88	4	20	1540.0	250.0	163.8	100	499.25	6	1200	6713.0	203.8	162.84	3.25	1640293.0	108
freq	41	129	48	106	33	6.0	19.0	3.0	23	4.00	111	16	19.0	19.0	19.00	20.00	19.0	

In [15]:
```
# Détermination des valeurs nulles:

sum(df.isnull().sum())
```

Out[15]: 0

Il n'y a aucune valeur nulle, ce qui nous permet de mener une analyse à l'aide un algorithme de ML.

Une présence de valeurs nulles (NaN) nous aurait obligés à les supprimer, car les résultats du ML auraient été faussés.

Pour une meilleure copréhension des variables, changeons leurs noms:

In [16]:
```
new_col_names = ['START YEAR',
 'START QUARTER',
 'COMPLETION YEAR',
 'COMPLETION QUARTER',
 'Project_locality',
 'Floor_area',
 'Lot_area',
 'TPECC_PBP',
 'PECC_PBP',
 'EPECC_PBP_SBY',
 'Construction_duration',
 'Unit_price_BP_m²',
 'Number_of_bulding_permits',
 'BSI_SBY',
 'WPI_BM_SBY',
 'TFA_BM',
 'Cumulative_liquidity',
 'PSI',
 'LPI_SBY',
 'NLE_by_banks',
 'ALE_by_banks',
 'Interest_rate',
 'ACC_PS_TC',
 'ACC_PS_BC',
 'Exchange_rate_dollar',
 'Non_official_exchange_rate',
 'CPI',
 'CPI_water_housing_fuel',
 'Stock_market_index',
 'Tehran_population',
 'Gold_price_per_ounce',
 'Number_of_bulding_permits_1',
 'BSI_SBY_1',
 'WPI_BM_SBY_1',
 'TFA_BM_1',
 'Cumulative_liquidity_1',
 'PSI_1',
 'LPI_SBY_1',
 'NLE_by_banks_1',
 'ALE_by_banks_1',
 'Interest_rate_1',
 'ACC_PS_TC_1',
```

```
'Number_of_bulding_permits_2',
'BSI_SBY_2',
'WPI_BM_SBY_2',
'TFA_BM_2',
'Cumulative_liquidity_2',
'PSI_2',
'LPI_SBY_2',
'NLE_by_banks_2',
'ALE_by_banks_2',
'Interest_rate_2',
'ACC_PS_TC_2',
'ACC_PS_BC_2',
'Exchange_rate_dollar_2',
'Non_official_exchange_rate_2',
'CPI_2',
'CPI_water_housing_fuel_2',
'Stock_market_index_2',
'Tehran_population_2',
'Gold_price_per_ounce_2',
'Number_of_bulding_permits_3',
'BSI_SBY_3',
'WPI_BM_SBY_3',
'TFA_BM_3',
'Cumulative_liquidity_3',
'PSI_3',
'LPI_SBY_3',
'NLE_by_banks_3',
'ALE_by_banks_3',
'Interest_rate_3',
'ACC_PS_TC_3',
'ACC_PS_BC_3',
'Exchange_rate_dollar_3',
'Non_official_exchange_rate_3',
'CPI_3',
'CPI_water_housing_fuel_3',
'Stock_market_index_3',
```

```
 'Exchange_rate_dollar_4',
 'Non_official_exchange_rate_4',
 'CPI_4',
 'CPI_water_housing_fuel_4',
 'Stock_market_index_4',
 'Tehran_population_4',
 'Gold_price_per_ounce_4',
 'Actual_sales_price_(output)',
 'Actual_construction_costs_(output)']
```

In [17]:
```
# Application des nouveaux noms de variables à notre dataframe:

df.set_axis(new_col_names, axis = 'columns', inplace = True)
```

In [18]:
```
df.head()
```

Out[18]:

	START YEAR	START QUARTER	COMPLETION YEAR	COMPLETION QUARTER	Project_locality	Floor_area	Lot_area	TPECC_PBP	PECC_PBP	EPECC_PBP_SBY	Construction_duration
2	81	1	85	1	1	3150	920	598.5	190	1010.84	16
3	84	1	89	4	1	7600	1140	3040	400	963.81	23
4	78	1	81	4	1	4800	840	480	100	689.84	15
5	72	2	73	2	1	685	202	13.7	20	459.54	4
6	87	1	90	2	1	3000	800	1230	410	631.91	13

In [19]:
```
# Maintenant, déterminons le type des variables présentes dans le dataframe:

df.dtypes
```

Out[19]:
```
START YEAR                          object
START QUARTER                       object
COMPLETION YEAR                     object
COMPLETION QUARTER                  object
Project_locality                    object
                                     ...
Stock_market_index_4                object
Tehran_population_4                 object
Gold_price_per_ounce_4              object
Actual_sales_price_(output)         object
Actual_construction_costs_(output)  object
Length: 109, dtype: object
```

In [20]:
```
df.dtypes.unique()
```

Out[20]:
```
array([dtype('O')], dtype=object)
```

Nous avons seulement un seul type de variable ('object'). Ce type n'est pas indiqué pour les algorithmes de ML.

Il nous faut donc changer et convertir en un type numérique ('int64'):

In [21]:
```
df = df.astype('int64')
```

In [22]:
```
df.dtypes
```

Out[22]:
```
START YEAR                          int64
START QUARTER                       int64
COMPLETION YEAR                     int64
COMPLETION QUARTER                  int64
Project_locality                    int64
                                    ...
Stock_market_index_4                int64
Tehran_population_4                 int64
```

```
Gold_price_per_ounce_4                int64
Actual_sales_price_(output)           int64
Actual_construction_costs_(output)    int64
```

In [23]:
```
df.dtypes.unique()
```

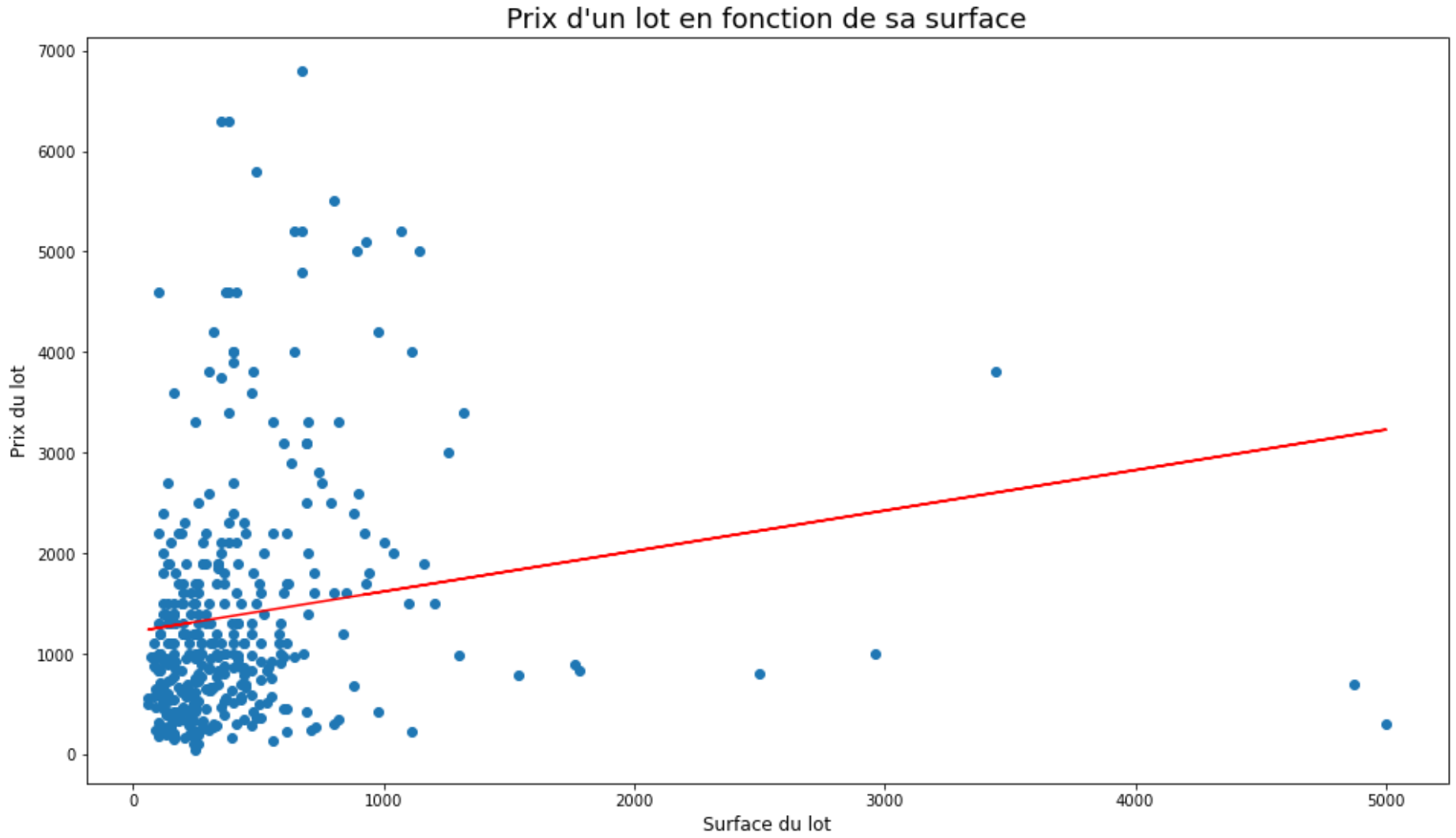
Out[23]:
```
array([dtype('int64')], dtype=object)
```

Nous avons maintenant un type de variable réglé à int64.

Avant de passer à l'analyse prédictive, faisons quelques visualisations sur notre dataset afin d'avoir une meilleure compréhension des données à notre disposition:

Commençons par visualiser la relation entre le prix d'un lot immobilier et sa surface:

In [24]:
```
plt.figure(figsize = (16,9))
plt.scatter(df['Lot_area'], df['Actual_sales_price_(output)'])
m, b = np.polyfit(df['Lot_area'], df['Actual_sales_price_(output)'], 1)
plt.plot(df['Lot_area'], m*(df['Lot_area'])+b, color = 'red')
plt.title('Prix d\'un lot en fonction de sa surface', fontsize = 18)
plt.xlabel('Surface du lot', fontsize = 12)
plt.ylabel('Prix du lot', fontsize = 12);
```



Nous remarquons que la relation entre la surface et le prix n'est pas forcément linéaire.

D'ores et déjà, nous pouvons présumer que la surface à elle seule

n'explique pas le prix de vente du lot

Appliquons un rapide test statistique afin de vérifier ou d'infirmer cette hypothèse.

In [25]:
```
# Appliquons le test de Pearson (vu que nous avons affaire à des variables continues)
```

In [26]:
```
from scipy.stats import pearsonr
```

In [27]:
```
pd.DataFrame(pearsonr(df['Lot_area'], df['Actual_sales_price_(output)']), index = ['Pearson_coeff', 'P_value'],
             columns = ['Test_result'])
```

Out[27]:

	Test_result
Pearson_coeff	0.163542
P_value	0.001551

Le test confirme bien une corrélation extrêmement faible (coefficient de pearson à 0.16).

Enchainons par une visualisation illustrant l'évolution des prix dans le temps:

In [28]:
```
df['START YEAR'].unique()
```

Out[28]:
```
array([81, 84, 78, 72, 87, 88, 76, 80, 75, 85, 86, 83, 74, 82, 77, 73, 79],
      dtype=int64)
```

In [29]:
```
plt.figure(figsize = (16,9))
sns.set_theme(style="whitegrid")
ax = sns.boxplot('START YEAR', 'Actual_sales_price_(output)', data = df)
ax.set_xlabel('Année de construction', fontsize = 16)
ax.set_ylabel('Prix du lot', fontsize = 16)
ax.set_title('Evolution des prix des lots suivant l\'année', fontsize = 20);
```

```
C:\Users\pedro_000\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other argume
nts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Evolution des prix des lots suivant l'année

Nous remarquons que les prix des lots augmentent de façon continue entre 1972 et 1988.

Nous pouvons vérifier cela à l'aide d'un test statistique (Pearson).

In [30]:
```
pd.DataFrame(pearsonr(df['START YEAR'], df['Actual_sales_price_(output)']), index = ['Pearson_coeff', 'P_value'],
             columns = ['Test_result'])
```

Out[30]:

	Test_result
Pearson_coeff	6.071236e-01
P_value	7.740337e-39

In [31]:
```
pearsonr(df['START YEAR'], df['Actual_sales_price_(output)'])
```

Out[31]:
```
(0.6071235812012881, 7.74033663478861e-39)
```

Le P de pearson est élevé (0.60), avec une P_value < 0.05, où 0.05 est le risque de 1ère espèce (seuil à 95%).

La relation entre le prix et l'année est statistiquement significative.

Il est temps de passer au ML à l'aide de l'ACP (Analyse en Composantes Principales), ou PCA en anglais.

C'est une méthode d'analyse de données utile afin de déterminer quelles sont les variables les plus influentes dans un jeu de données. Elle consiste à transformer des variables liées entre elles en de

La variable que l'on veut expliquer ici est "Actual_sales*price*(output)". Commençons par l'isoler, puis la supprimer du nouveau dataframe à utiliser:

In [32]:
```
price = df['Actual_sales_price_(output)']
```

In [33]:
```
df_1 = df.drop('Actual_sales_price_(output)', axis = 1)
```

Supprimons aussi la variable "Actual_construction*costs*(output)" qui est une autre variable à expliquer, mais dont nous n'aurons pas besoin pour le moment:

In [34]:
```
df_1 = df_1.drop('Actual_construction_costs_(output)', axis = 1)
```

Comme il y'a 5 séries temporelles et afin de faciiter la lecture des résultats de l'ACP, il est judicieux de n'en garder qu'une seule:

In [35]:
```
df_1 = df_1.iloc[:,:31]
```

In [36]:
```
df_1.head()
```

Out[36]:

START YEAR	START QUARTER	COMPLETION YEAR	COMPLETION QUARTER	Project_locality	Floor_area	Lot_area	TPECC_PBP	PECC_PBP	EPECC_PBP_SBY	Construction_duration

	START YEAR	START QUARTER	COMPLETION YEAR	COMPLETION QUARTER	Project_locality	Floor_area	Lot_area	TPECC_PBP	PECC_PBP	EPECC_PBP_SBY	Construction_duration
2	81	1	85	1	1	3150	920	598	190	1010	16
3	84	1	89	4	1	7600	1140	3040	400	963	23
4	78	1	81	4	1	4800	840	480	100	689	15

Maintenant, il faut normaliser nos données contenues dans df_1: ceci est primordial afin de réduire l'effet d'échelle, du fait de la différence des unités de mesure:

In [37]:
```
from sklearn.preprocessing import StandardScaler
```

In [38]:
```
scaler = StandardScaler()
```

In [39]:
```
N = scaler.fit_transform(df_1)
```

Appliquons maintenant l'ACP:

In [40]:
```
from sklearn.decomposition import PCA
```

In [41]:
```
pca = PCA()
```

In [42]:
```
coord = pca.fit_transform(N)
```

Affichons la part de la variance expliquée pour chaque composante de l'ACP:

In [43]:
```
print('La part de la variance expliquée est:', pca.explained_variance_)
```

```
La part de la variance expliquée est: [1.67024619e+01 3.11419601e+00 2.08634999e+00 1.70558544e+00
```

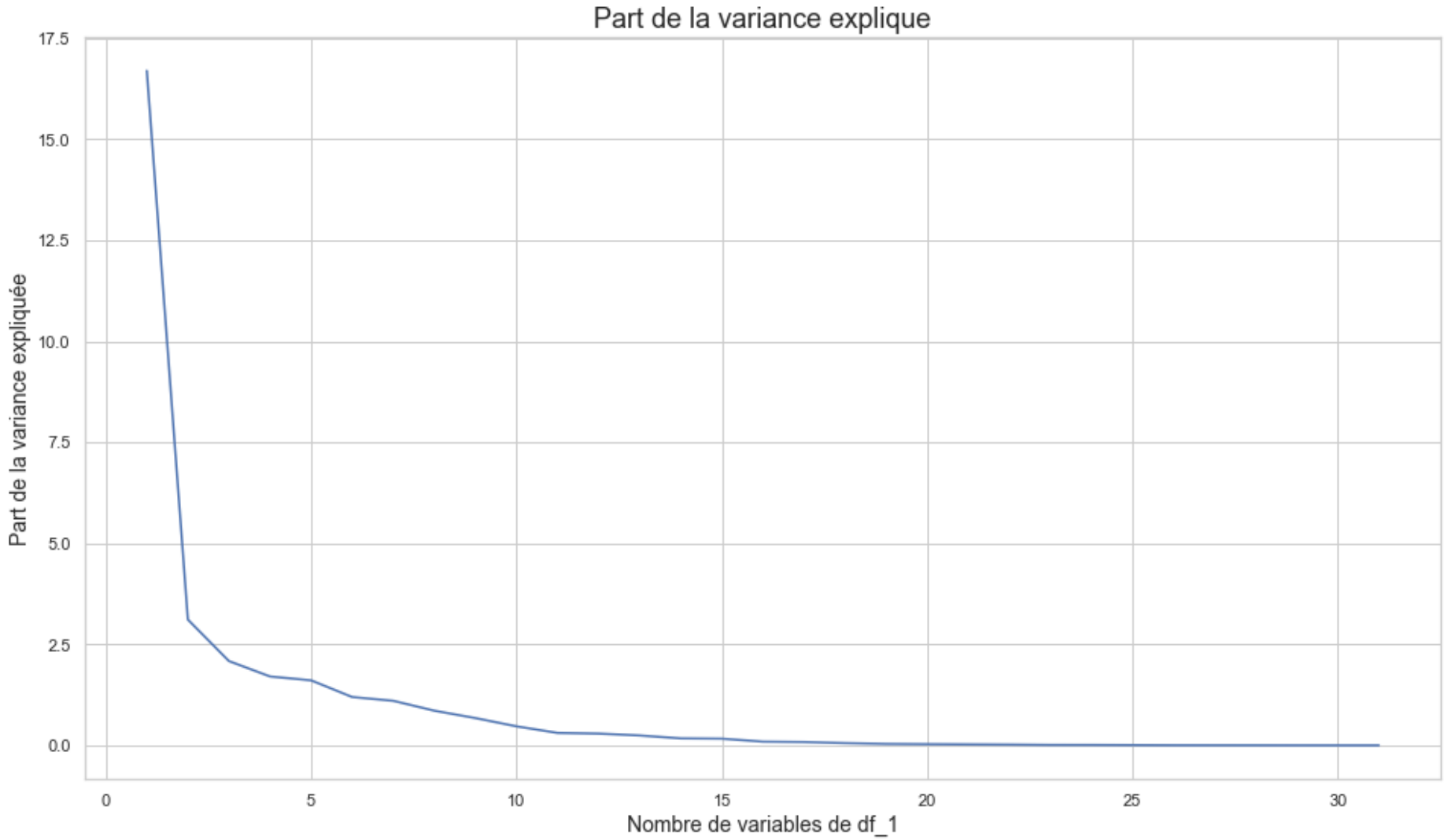
```
1.60973108e+00 1.19709554e+00 1.10297248e+00 8.58588180e-01
6.75693126e-01 4.70959160e-01 3.08466945e-01 2.91343883e-01
2.45383734e-01 1.72914377e-01 1.66526700e-01 9.23262066e-02
8.31560060e-02 5.56581409e-02 3.62320958e-02 2.96211253e-02
2.38618794e-02 1.89834966e-02 1.08573489e-02 1.00199516e-02
6.60496215e-03 3.10049129e-03 2.15020692e-03 1.87421318e-03
```

Traçons une courbe des valeurs ci-haut:

In [44]:
```
df_1.shape
```

Out[44]:
```
(372, 31)
```

In [45]:
```
plt.figure(figsize = (16,9))
plt.plot(np.arange(1, 32), pca.explained_variance_)
plt.xlabel('Nombre de variables de df_1', fontsize = 14)
plt.ylabel('Part de la variance expliquée', fontsize = 14)
plt.title('Part de la variance explique', fontsize = 18);
```

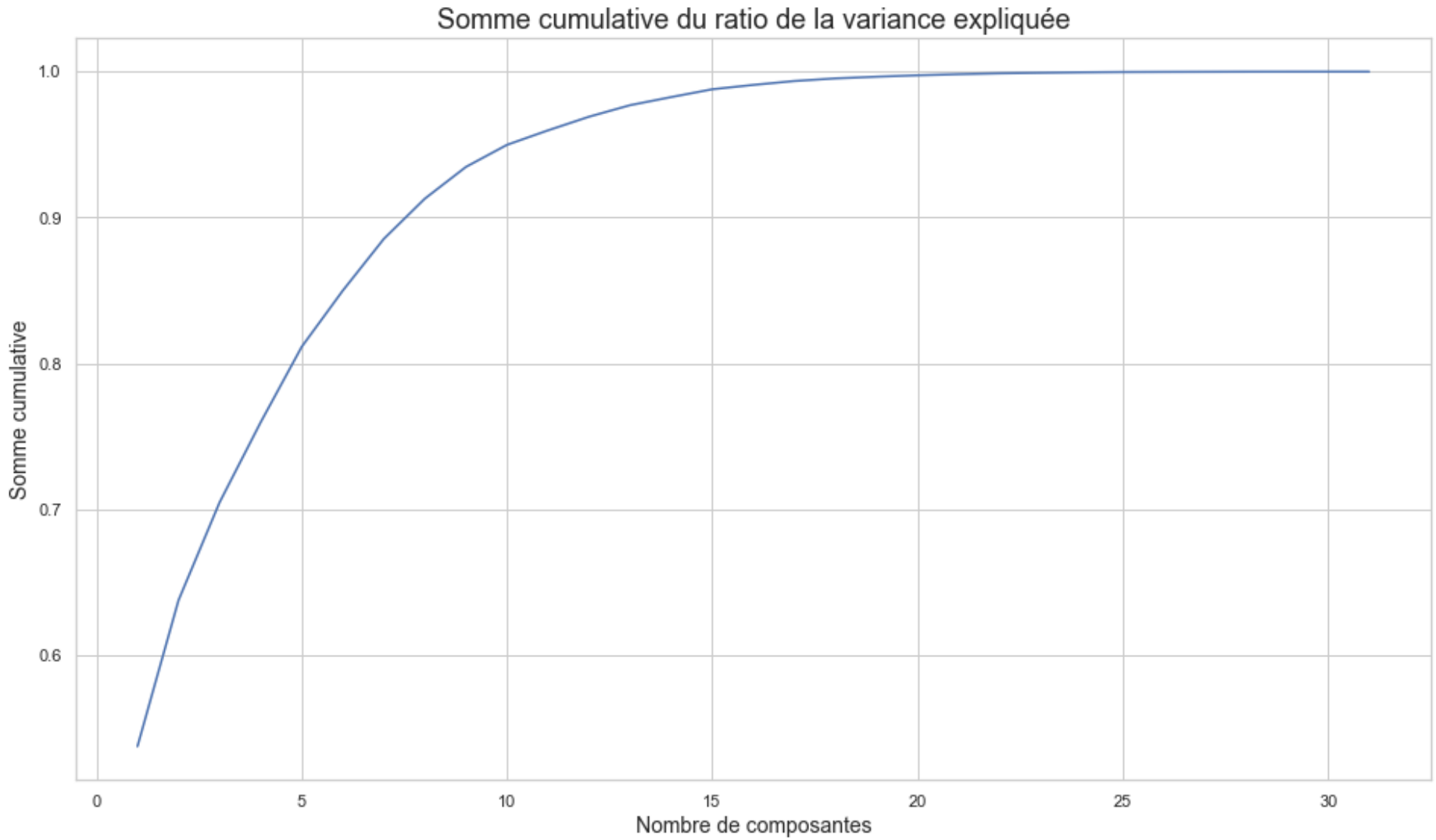


Maintenant, il nous faut déterminer le nombre de variables explicatives à retenir. Pour ce faire, nous allons appliquer la méthode dite "du coude": il s'agit de déterminer le point d'inflection de la courbe du ratio de la variance expliquée:

In [46]:
```
print('Le ration de la variance expliquée est de:', pca.explained_variance_ratio_)
```

```
Le ration de la variance expliquée est de: [5.37340737e-01 1.00187888e-01 6.71206943e-02 5.48709850e-02
 5.17872210e-02 3.85121788e-02 3.54841131e-02 2.76219402e-02
 2.17379596e-02 1.51513916e-02 9.92379783e-03 9.37292583e-03
 7.89432580e-03 5.56288882e-03 5.35738864e-03 2.97025864e-03
 2.67524092e-03 1.79059749e-03 1.16563541e-03 9.52951568e-04
 7.67668856e-04 6.10724700e-04 3.49295563e-04 3.22355363e-04
 2.12490544e-04 9.97469882e-05 6.91750577e-05 6.02959669e-05
 2.12672639e-05 5.86052981e-06 9.75392625e-33]
```

In [47]:
```
# Courbe de la somme cumulative du ratio de la variance expliquée:

plt.figure(figsize = (16,9))
plt.plot(np.arange(1,32), np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Nombre de composantes', fontsize = 14)
plt.ylabel('Somme cumulative', fontsize = 14)
plt.title('Somme cumulative du ratio de la variance expliquée', fontsize = 18);
```

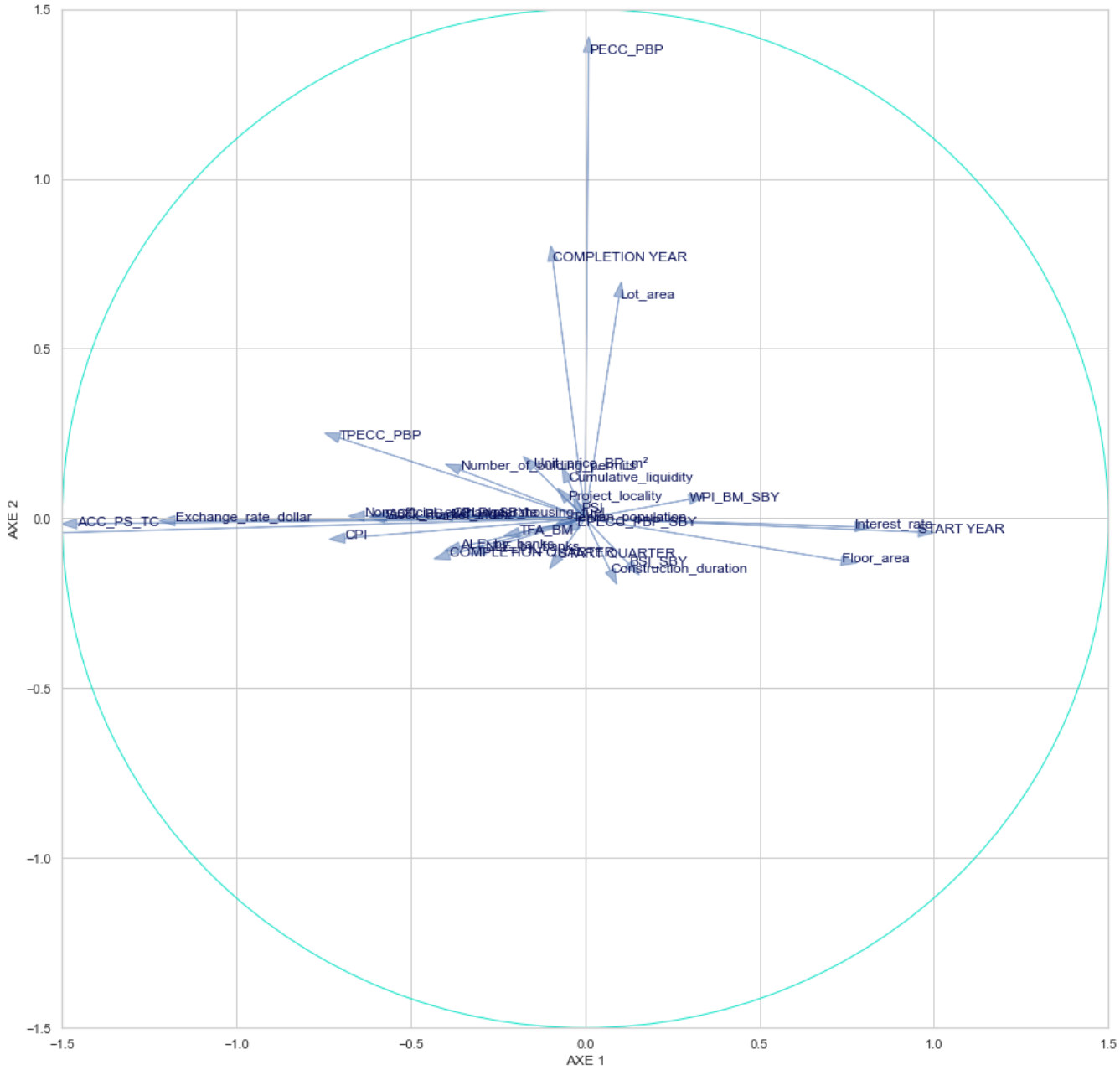


Approximativement, 70% de la variance expliquée l'est avec 3 axes. Nous retenons 3 composantes principales.

Maintenant, traçons le cercle des corrélations:

In [69]:

```
R = np.sqrt(pca.explained_variance_)

corvar = np.zeros((31, 31))
for k in range(31):
    corvar[:, k] = pca.components_[:, k] * R[k]


fig, axes = plt.subplots(figsize=(16, 16))
axes.set_xlim(-1.5, 1.5)
axes.set_ylim(-1.5, 1.5)

for j in range(31):
    plt.annotate(df_1.columns[j], (corvar[j, 0], corvar[j, 1]), color='#091158')
    plt.arrow(0, 0, corvar[j, 0], corvar[j, 1],
        alpha=0.5, head_width=0.03, color='b')

plt.plot([-1.5, 1.5], [0, 0], color='silver', linestyle='-', linewidth=1)
plt.plot([0, 0], [-1.5, 1.5], color='silver', linestyle='-', linewidth=1)

cercle = plt.Circle((0, 0), 1.5, color='#16E4CA', fill=False)
axes.add_artist(cercle)
plt.xlabel('AXE 1')
plt.ylabel('AXE 2')
plt.show()
```



Nous remarquons que le prix de vente est positivement corrélé au

taux t'intérêt (interest_rate), à l'année de départ du projet (START_YEAR) et à la surface (Floor_area).

Aussi, le prix est négativement corrélé au taux de change face au dollar américain (Exchange_rate_dollar), du coût moyen de construction dans le secteur privé au moment de la finalisation de la construction (ACC_PS_TC), au coût de construction estimé avant le début du chantier de construction (TPECC_PBP), ainsi que l'indice des prix à la consommation (CPI).

Ce genre d'analyse est très utile afin de comprendre quels sont les leviers qui influent le plus sur la variable prix.