Rapport final : Projet statistique

PREDICTION DE RECOLTE

Réalisé par :

- Soukaina CHAKOUR
- Abdelmoula ELBOUHALI
- Noemane EL HRIZI
- Taha LAKHDARI

Encadré par :

- Mme. Ikram CHAIRI
- Mr. Saad BENJELOUN

Table des matières

Introduction

Les recherches des sciences de statistiques et d'intelligence artificielle de nos jours s'intéressent de plus en plus au secteur d'agriculture vu qu'il constitue un moteur primordial de l'économie d'une pléthore de pays au monde. Dans ce même sens, les bonnes prévisions des rendements de cultures jouent un rôle majeur dans la prise de décision aux niveaux mondial, régional et local. En effet, ces prévisions permettent de comprendre mieux l'évolution des rendements agricoles en fonctions des différents paramètres et en différentes régions, ainsi des décisions de choix de cultures adéquates à chaque région peuvent être prises par exemple.

Ceci est exactement le cas de notre travail qui porte sur la prédiction de la récolte de différentes cultures en Inde en fonction de la géographie, la saison, et la surface cultivée. Pour ce faire, des tests statistiques, des modèles de machine learning, et des visualisations de données sont mises en œuvre.

I. Étude descriptive :

1. *Description des données :*

La data de notre projet est composée de 9 variables qui sont :

State	Nom de ‘état indien
District	Nom de la région dans l'État
Year	Année de culture
Season	La saison de culture
Crop	Nom de la culture
Area	Superficie cultivée
Production	Production de la culture
Rain full	La quantité de pluie en Inde
Température	Température moyenne dans la région

	State	District	Year	Season	Crop	Area	Production	Rainfall	Temperature
0	Bihar	ARARIA	1997	Autumn	Rice	17876.0	21397.0	1303.7	31.77
1	Bihar	BANKA	1997	Autumn	Rice	38.0	59.0	1303.7	31.77
2	Bihar	BEGUSARAI	1997	Autumn	Rice	7812.0	11698.0	1303.7	31.77
3	Bihar	BHAGALPUR	1997	Autumn	Rice	1215.0	1867.0	1303.7	31.77
4	Bihar	BHOJPUR	1997	Autumn	Rice	682.0	733.0	1303.7	31.77

2. *Diagnostic :*

De prime abord, on commence par un diagnostic de notre dataset afin de chercher les éventuelles anomalies.

Notre data est composée de 74975 observations.

```
data.shape
```

```
(74975, 9)
```

- ***Valeurs manquantes :***

On remarque que la dataset ne contient pas de valeurs manquantes.

```
data.isna().sum()
```

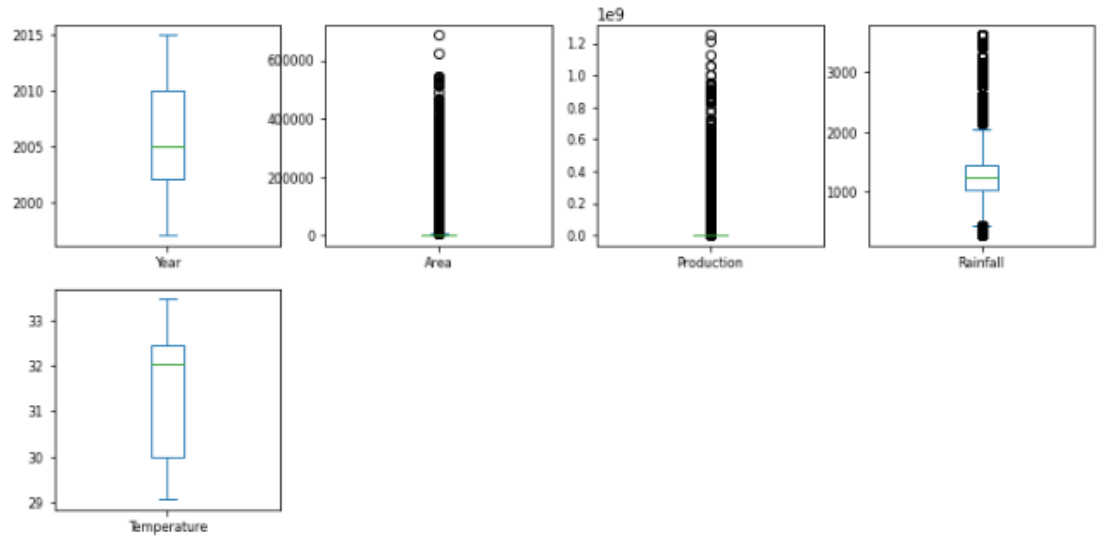
```
State          0
District       0
Year           0
Season         0
Crop           0
Area           0
Production     0
Rainfall       0
Temperature    0
dtype: int64
```

- ***Valeurs aberrantes :***

Pour la visualisation des valeurs aberrantes on utilise les Boxplots.

```
data.plot(kind='box', subplots=True, layout=(4,4), fontsize=8,figsize=(12,12))
```

```
Year              AxesSubplot(0.125,0.71587;0.168478x0.16413)
Area           AxesSubplot(0.327174,0.71587;0.168478x0.16413)
Production     AxesSubplot(0.529348,0.71587;0.168478x0.16413)
Rainfall       AxesSubplot(0.731522,0.71587;0.168478x0.16413)
Temperature      AxesSubplot(0.125,0.518913;0.168478x0.16413)
dtype: object
```

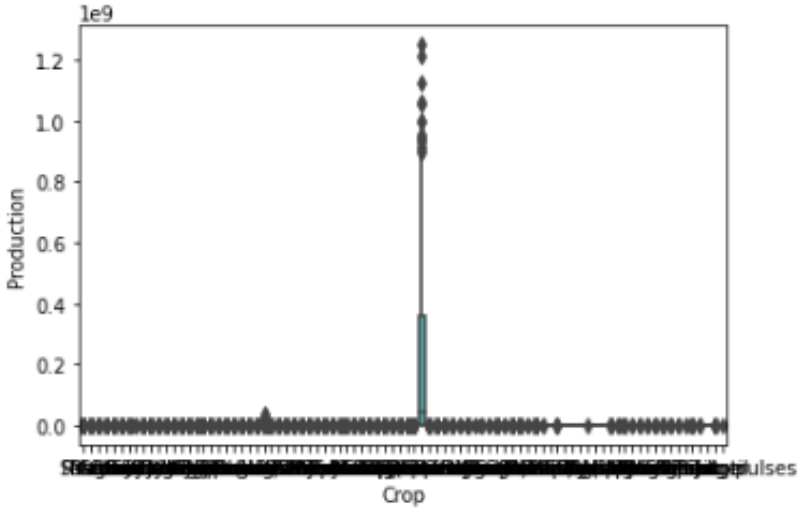


On remarque alors qu'il y a une grande dispersion des valeurs des deux variables « Production » et « Area ».

Cependant, d'après le graphe suivant, on conclut que cette dispersion est liée, pour « Production », à la nature du « Crop » et qu'il ne s'agit pas donc de valeurs aberrantes.

```
sns.boxplot(x=data["Crop"],y=data["Production"])
```

```
<AxesSubplot:xlabel='Crop', ylabel='Production'>
```



3. Variables qualitatives :

Cette partie nous permet d'avoir une description détaillée de toutes les variables qualitatives.

On commence alors par l'exploration du nombre de valeurs uniques de chacune de ces variables.

```
for col in data.select_dtypes('object'):
  print(f'{col :-<15} {data[col].nunique()}')
```

```
State---------- 12
District------- 253
Season--------- 6
Crop----------- 86
```

On remarque ainsi que les deux variables « District » et « Crop » ont un grand nombre de valeurs uniques. Ceci nous laisse noter qu'une factorisation de la variable « District » lors de la construction des modèles de prédiction engendrera un grand nombre de colonnes.

On découvre aussi la dataset contient 6 saisons différemment à la normale. Alors, on doit nécessairement afficher toutes les valeurs uniques de chaque colonne, ce qu'est montré dans la figure ci-dessous.

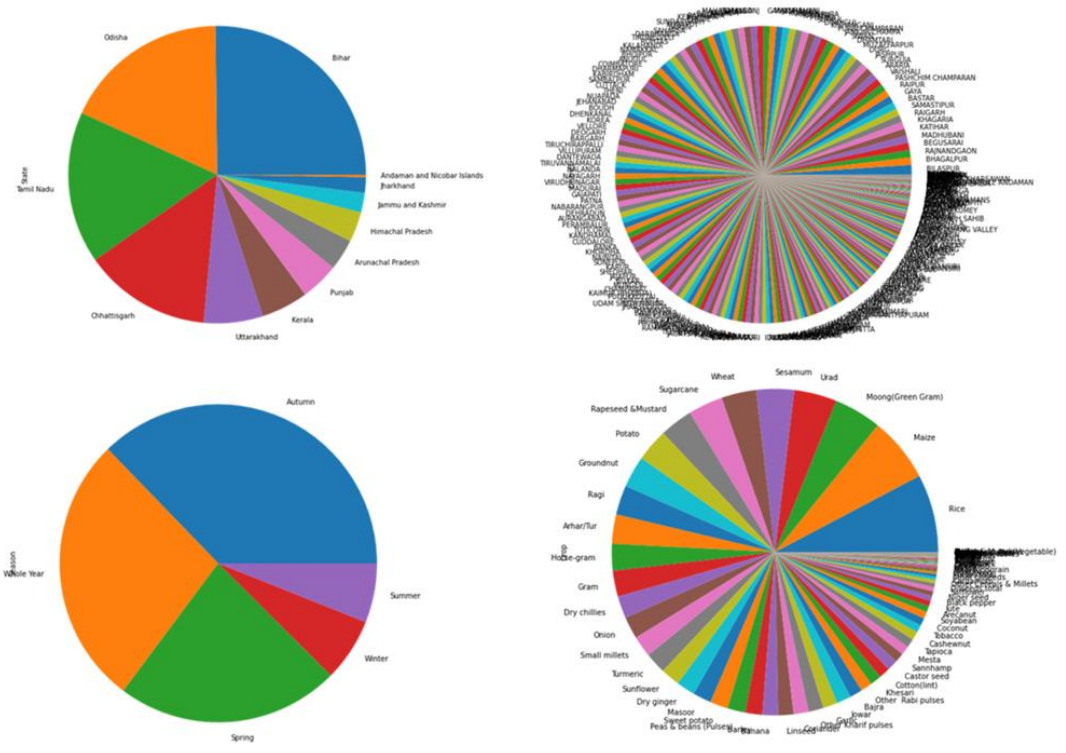
```
for col in data.select_dtypes('object'):
  print(f'{col :-<10} {data[col].unique()}')
```

```
State----- ['Bihar' 'Odisha' 'Arunachal Pradesh' 'Jammu and Kashmir' 'Punjab'
 'Tamil Nadu' 'Kerala' 'Himachal Pradesh' 'Andaman and Nicobar Islands'
 'Chhattisgarh' 'Uttarakhand' 'Jharkhand']
District-- ['ARARIA' 'BANKA' 'BEGUSARAI' 'BHAGALPUR' 'BHOJPUR' 'DARBHANGA' 'GAYA'
 'GOPALGANJ' 'JEHANABAD' 'KAIMUR (BHABUA)' 'KATIHAR' 'KHAGARIA'
 'KISHANGANJ' 'MADHEPURA' 'MADHUBANI' 'MUNGER' 'MUZAFFARPUR' 'NAWADA'
 'PASHCHIM CHAMPARAN' 'PATNA' 'PURBI CHAMPARAN' 'PURNIA' 'ROHTAS'
 'SAHARSA' 'SAMASTIPUR' 'SARAN' 'SHEOHAR' 'SITAMARHI' 'SIWAN' 'SUPAUL'
 'VAISHALI' 'ANUGUL' 'BALANGIR' 'BALESHWAR' 'BARGARH' 'BHADRAK' 'BOUDH'
 'CUTTACK' 'DEOGARH' 'DHENKANAL' 'GAJAPATI' 'GANJAM' 'JAGATSINGHAPUR'
 'JAJAPUR' 'JHARSUGUDA' 'KALAHANDI' 'KANDHAMAL' 'KENDRAPARA' 'KENDUJHAR'
 'KHORDHA' 'KORAPUT' 'MALKANGIRI' 'MAYURBHANJ' 'NABARANGPUR' 'NAYAGARH'
 'NUAPADA' 'PURI' 'RAYAGADA' 'SAMBALPUR' 'SONEPUR' 'SUNDARGARH'
 'CHANGLANG' 'DIBANG VALLEY' 'EAST KAMENG' 'EAST SIANG' 'LOHIT'
 'LOWER SUBANSIRI' 'PAPUM PARE' 'TAWANG' 'TIRAP' 'UPPER SIANG'
 'UPPER SUBANSIRI' 'WEST KAMENG' 'WEST SIANG' 'AURANGABAD' 'BUXAR' 'JAMUI'
 'LAKHISARAI' 'NALANDA' 'SHEIKHPURA' 'ANANTNAG' 'BADGAM' 'BARAMULLA'
 'DODA' 'JAMMU' 'KATHUA' 'KUPWARA' 'POONCH' 'PULWAMA' 'RAJAURI' 'SRINAGAR'
 'UDHAMPUR' 'AMRITSAR' 'BATHINDA' 'FARIDKOT' 'FATEHGARH SAHIB' 'FIROZEPUR'

Season---- ['Autumn      ' 'Kharif      ' 'Rabi        ' 'Summer      ' 'Whole Year '
 'Winter      ']
```

On trouve effectivement qu'il y a une irrégularité de saisie dans la colonne « Season », (ex : l'automne est saisi tantôt comme 'Autumn' tantôt comme 'Kharif'). Cette irrégularité doit être nécessairement réglée dans la partie de nettoyage de la data.

Ensuite, on affiche sous forme de camemberts la distribution des valeurs unique de chacune des colonnes qualitatives.



Ainsi, on tire les remarques suivantes :

* Les régions les plus agricoles sont Bihar, Odisha, Tamil Nadu, Chhattisgarh.
* La culture la plus fréquente en Inde est le Riz.

4. *Analyse descriptive et univariée des variables quantitatives :*

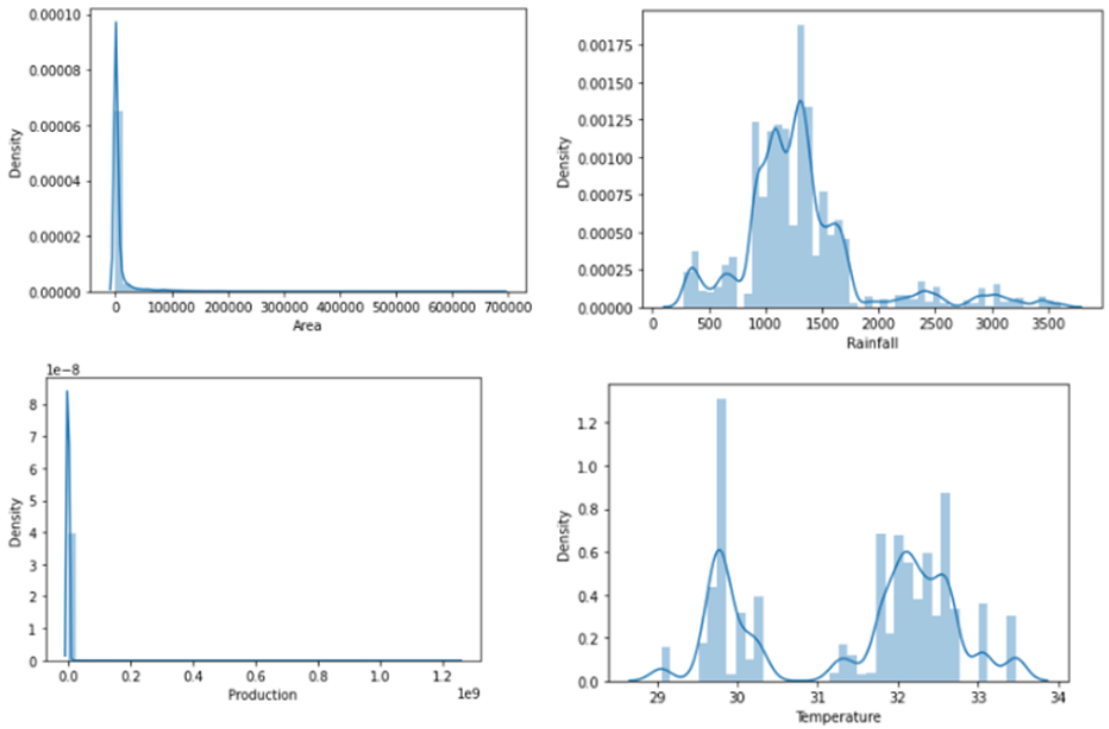
La figure ci-dessous représente le sommaire statistique des variables quantitatives de notre dataset.

	Year	Area	Production	Rainfall	Temperature
count	74975.000000	74975.000000	7.497500e+04	74975.000000	74975.000000
mean	2005.588169	8360.384978	1.494769e+06	1302.775097	31.456754
std	4.964027	30176.807737	2.895768e+07	555.243785	1.244797
min	1997.000000	0.200000	1.000000e-02	274.700000	29.050000
25%	2002.000000	68.000000	6.440000e+01	1032.400000	29.990000
50%	2005.000000	472.000000	5.260000e+02	1247.000000	32.050000
75%	2010.000000	2882.000000	4.152000e+03	1437.300000	32.450000
max	2015.000000	687000.000000	1.250800e+09	3616.700000	33.470000

Depuis ce sommaire, on peut extraire les indices statistiques: de position (moyenne) et de dispersion (variance). Pour notre variable dépendante « Production », la moyenne est de $1{,}5\times10^{6}$, et la variance (le carré de std) est de $8{,}4\times10^{14}$ qui témoigne la grande dispersion de la variable « Production » (un fait déjà observé à l'aide de la boite à moustaches).

On conclut de ce sommaire aussi que la variable « Température » ne varie pas significativement vu que son écart-type est très faible de *1,2*. Ainsi, cette variable n'aurait pas d'influence significative sur nos modèles de prédiction.

Les graphes ci-dessous visualisent la fonction de densité de chacune des variables da la data.



5. *Nettoyage des données :*

La seule anomalie observée dans notre dataset est l'irrégularité de saisie dans les valeurs de la colonne qualitative « Season ». Pour la fixer, on utilise le code suivant :

```
# cortrection de l'irregulatité de saisie
Season_map = {'Autumn     ': 'Autumn', 'Kharif     ': 'Autumn', 'Summer     ': 'Summer', 'Rabi       ':
              'Spring', 'Whole Year ': 'Whole Year', 'Winter     ': 'Winter'}
def mapper(Season):
  return Season_map[Season]
data = pd.DataFrame(data)
data['Season'] = data['Season'].apply(mapper)
data['Season'].unique()
```

```
array(['Autumn', 'Spring', 'Summer', 'Whole Year', 'Winter'], dtype=object)
```

6. Analyse multivariée des données :

- ***Production totale pour chaque Crop:***

Dans notre dataset, il y a 86 Crops mais on voit bien d'après ce graphe que la production de la grande partie d'eux est très minime par rapport à 4 Crops qui sont les plus dominants en production, ainsi on se concentrera après que sur ces quatre.

Sélection des Top Crops :

Alors, on travaille désormais avec les quatre meilleurs Crops, qui sont : Coconut, Sugarcane, Rice et Wheat.

```
crops=['Coconut ', 'Sugarcane', 'Rice', 'Wheat']
data1=data[data.Crop.isin(crops)]
data1
```

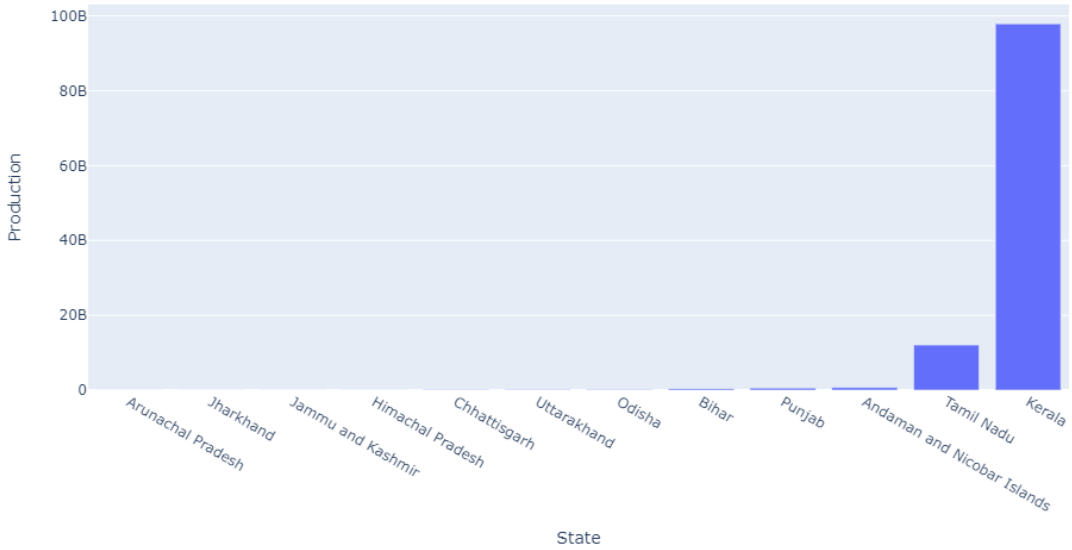
	State	District	Year	Season	Crop	Area	Production	Rainfall	Temperature
0	Bihar	ARARIA	1997	Autumn	Rice	17876.0	21397.0	1303.7	31.77
1	Bihar	BANKA	1997	Autumn	Rice	38.0	59.0	1303.7	31.77
2	Bihar	BEGUSARAI	1997	Autumn	Rice	7812.0	11698.0	1303.7	31.77
3	Bihar	BHAGALPUR	1997	Autumn	Rice	1215.0	1867.0	1303.7	31.77
4	Bihar	BHOJPUR	1997	Autumn	Rice	682.0	733.0	1303.7	31.77
...	...	...	...	...	...	...	...	...	...
74955	Odisha	SAMBALPUR	2015	Winter	Rice	91000.0	98000.0	1210.1	29.90
74956	Odisha	SONEPUR	2015	Winter	Sugarcane	1.0	52.0	1210.1	29.90
74964	Odisha	SONEPUR	2015	Winter	Rice	86000.0	209000.0	1210.1	29.90
74968	Odisha	SUNDARGARH	2015	Winter	Sugarcane	17.0	1173.9	1210.1	29.90
74974	Odisha	SUNDARGARH	2015	Winter	Rice	114000.0	156000.0	1210.1	29.90

11513 rows × 9 columns

- ***Production totale par State:***

La visualisation de la production totale en fonction de la région nous permet de déterminer les régions qui ont les plus grandes productions.

```
temp = data.groupby(by='State')['Production'].sum().reset_index().sort_values(by='Production')
px.bar(temp, 'State', 'Production')
```



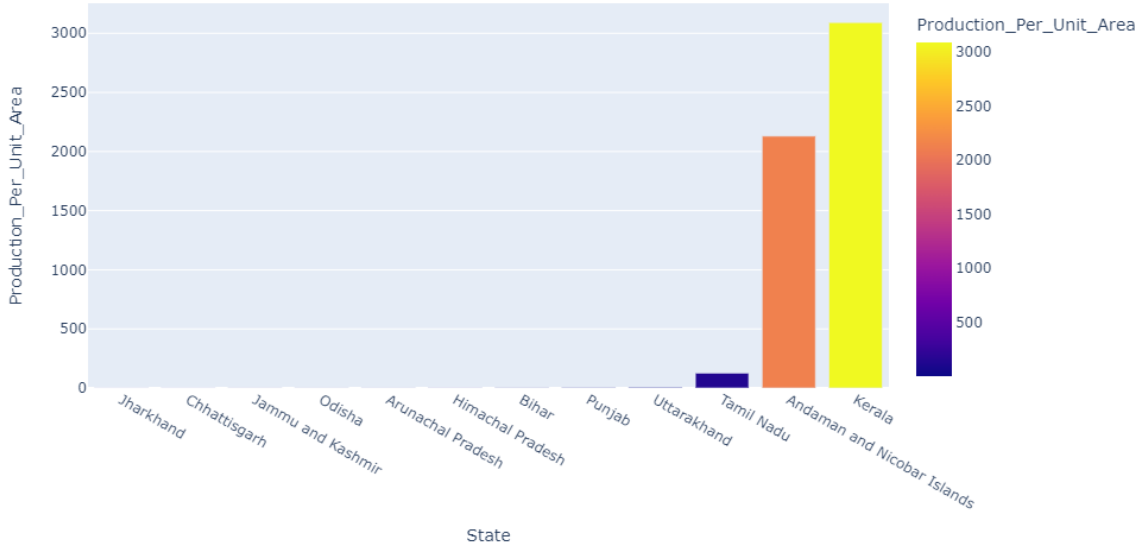
On voie très bien que les régions qui ont la plus grande production en Inde sont Kerala et Tamil Nadu.

- ***<u>Productivité par Surface Cultivée pour chaque State:</u>***

Dans cette partie on vise à calculer la productivité de chaque région par unité de surface. Alors ; on calcule la productivité par la relation suivante :

$$Productivité = \frac{Production}{Superficie\ récolté}$$

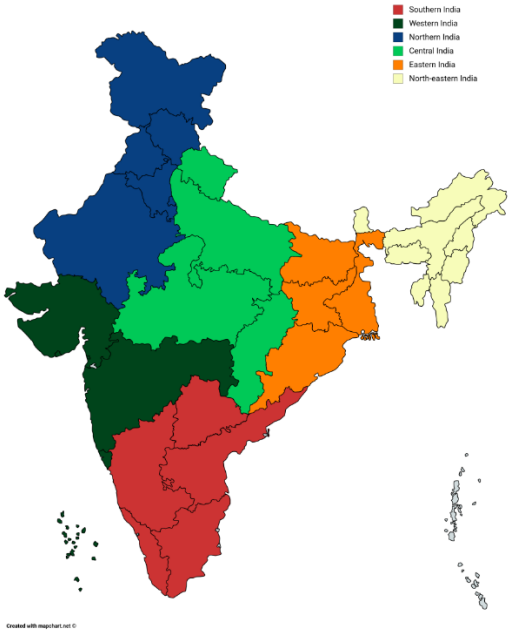
```
temp = data.groupby('State')['Area', 'Production'].sum().reset_index()
temp['Production_Per_Unit_Area'] = temp['Production']/temp['Area']
temp = temp.sort_values(by='Production_Per_Unit_Area')
px.bar(temp, 'State', 'Production_Per_Unit_Area', color='Production_Per_Unit_Area')
```



On remarque que les régions qui ont la plus grande productivité sont Kerala, Andaman and Nicobar Islands et Tamil Nadu. On voit donc que Andaman and Nicobar Islands avait une production cumulative bien inférieure à celle de Tamil Nadu alors que c'est exactement l'inverse en terme de productivité.

- ***Production totale par Zone géographique***

Pour rendre la visualisation de la relation entre Production et facteur géographique plus informatif, nous avons pensé à voir l'emplacement géographique des différentes States indiennes existantes dans notre data. En effet, l'Inde est subdivisée six zones géographiques comprenant chacune 28 États.



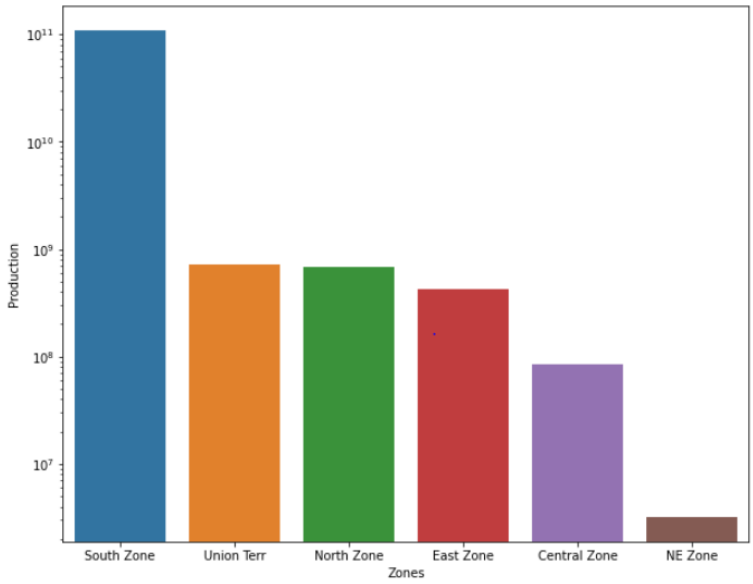
Ainsi, nous avons attribué chaque State à sa zone correspondante.

```
north_india = ['Jammu and Kashmir', 'Punjab', 'Himachal Pradesh', 'Uttarakhand']
east_india = ['Bihar', 'Odisha', 'Jharkhand']
south_india = ['Kerala' ,'Tamil Nadu']
north_east_india = ['Arunachal Pradesh']
central_india = ['Chhattisgarh']
ut_india = ['Andaman and Nicobar Islands']
def get_zonal_names(row):
    if row['State'].strip() in north_india:
        val = 'North Zone'
    elif row['State'].strip()  in south_india:
        val = 'South Zone'
    elif row['State'].strip()  in east_india:
        val = 'East Zone'

    elif row['State'].strip()  in central_india:
        val = 'Central Zone'
    elif row['State'].strip()  in north_east_india:
        val = 'NE Zone'
    elif row['State'].strip()  in ut_india:
        val = 'Union Terr'
    else:
        val = 'No Value'
    return val

data1['Zones'] = data1.apply(get_zonal_names, axis=1)
data1['Zones'].unique()
```

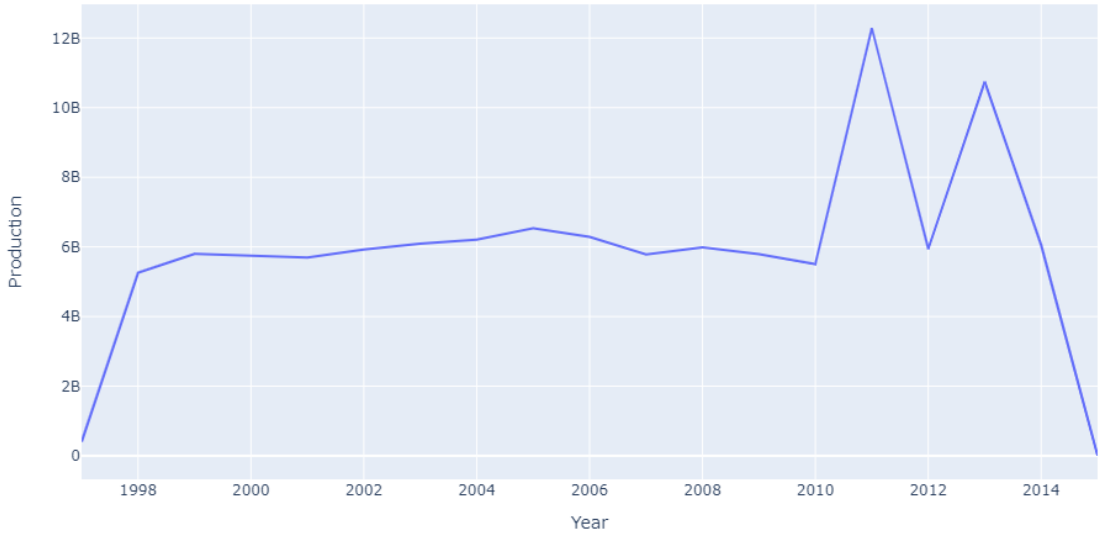
Ainsi, nous visualisons la production totale de nos 4 Crops dans chaque zone :



On voit donc que la zone sud est en tête en termes de production globale alors qu'elle ne correspond dans notre data qu'à deux États (Kerala et Tamil Nadu). Elle est suivie aussi par la zone Union Terr qui n'est que des petites îles (Andaman and Nicobar Islands).

- ***Evolution de la production annuelle :***

On visualise l'évolution temporelle de la production agricole en Inde.



La production agricole de l'Inde reste stagnante durant tout cet intervalle de temps, sauf pour les deux années 2011 et 2013 qui ont connu des pics de production.

II. Etude statistique :

1. *Loi de distribution de variable dépendante « Production » :*

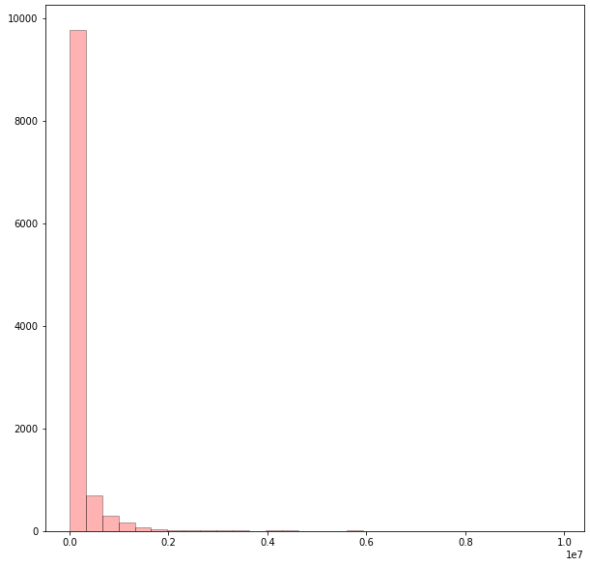
Introduction :

La modélisation de la distribution de données (probability distribution fitting, ou distribution fitting en anglais) est le fait de trouver les paramètres de la loi de distribution de probabilité (ou de plusieurs lois candidates) qui correspond aux données que l'on cherche à modéliser.
La modélisation de la distribution de données est une tâche qui peut s'effectuer de deux manières :

- Manuellement. On a une idée de la loi de distribution, ou de quelques lois candidates. Pour chaque loi, On trouve les paramètres optimaux par maximum de vraisemblance (ou Maximum Likelihood Estimation, MLE en anglais), ou par la méthode des moments (Method of Moments) par exemple.
- Automatiquement. On utilise une librairie ou un logiciel spécialisé, qui a déjà implémenté les maximums de vraisemblance de nombreuses lois, et cherchez à trouver la meilleure loi et les meilleurs paramètres d'un coup.

Dans notre cas on utilise la méthode automatique en utilisant La bibliothèque « fitter » dans pyhton qui fournit des méthodes simples permettant d'identifier la distribution qui modélise le mieux une distribution de données. Il utilise 80 distributions de scipy et permet de tracer les résultats pour vérifier la distribution la plus probable et les meilleurs paramètres.

On commence par visualiser l'histogramme de la distribution de données pour notre variable recherchée « Production » :



On remarque une grande inclinaison de la distribution des valeurs de production au côté gauche de l'histogramme (skewness), on ne peut même pas visualiser les autres valeurs de l'histogramme à cause de la grande différence entre la distribution.

On décide donc, pour mieux visualiser, de diviser la distribution en crops ; Les productions respectivement des cultures « Coconut », « Sugarcane », « Rice » et « Wheat ». On utilise maintenant la bibliothèque « fitter » afin d'identifier la distribution qui modélise le mieux chacune des quatre distributions de données de production.

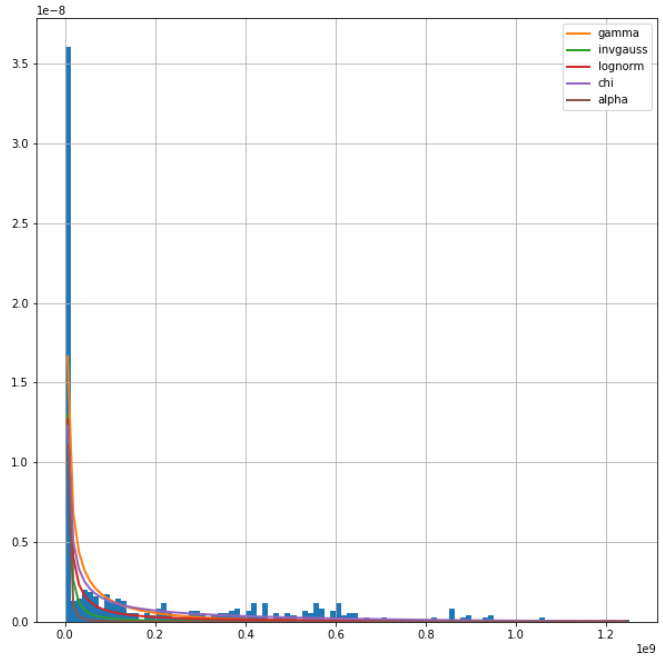
Les lois utilisés (à tester) sont :

```
['norm', 'beta','gamma', 'pareto', 't', 'lognorm', 'invgamma', 'invgauss', 'loggamma', 'alpha', 'chi', 'chi2']
```

Ces lois correspondent globalement aux lois les plus communes, implémentées dans scipy. En sortie on aura les 5 meilleures lois qui modélisent le mieux nos données parmi les distributions de lois utilisés. Les lois sont classées par « somme de résidus au carrés ».

- <u>Pour la distribution des productions de « Coconut » :</u>

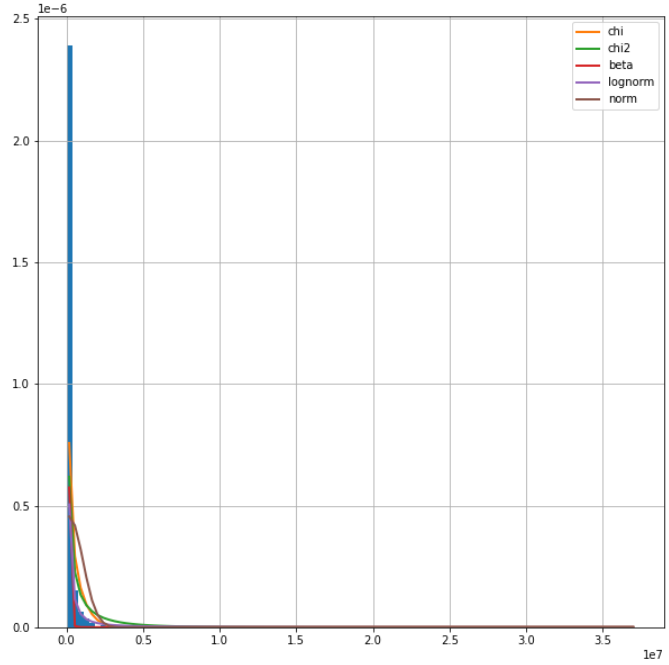
	sumsquare_error	aic	bic	kl_div
gamma	4.311337e-16	4721.947020	-23380.227883	inf
invgauss	5.610735e-16	4920.620405	-23232.441583	inf
lognorm	5.699843e-16	4597.052006	-23223.601974	inf
chi	5.926689e-16	4466.070607	-23201.707777	inf
alpha	6.619009e-16	5467.690434	-23139.728527	inf



D’après ces résultats qui précèdent, les lois qui modélisent le plus la distribution de production de « Coconut » : est la loi gamma.

- <u>Pour la distribution des productions de « Sugarcane » :</u>

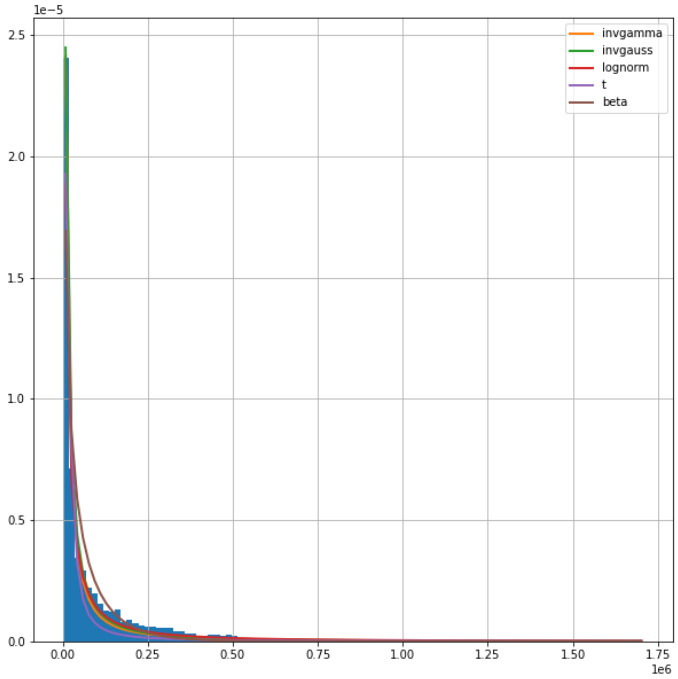
	sumsquare_error	aic	bic	kl_div
chi	2.704365e-12	34678.891453	-393558.845965	inf
chi2	3.163371e-12	4724.521022	-391841.887413	inf
beta	3.326724e-12	60875.354999	-391281.153338	inf
lognorm	3.559013e-12	4233.551432	-390551.248617	inf
norm	3.925067e-12	61806.942192	-389488.346085	inf



D’après ces résultats qui précèdent, les lois qui modélisent le plus la distribution de production de « Sugarcane » est la loi chi.

- Pour la distribution des productions de « Rice » :

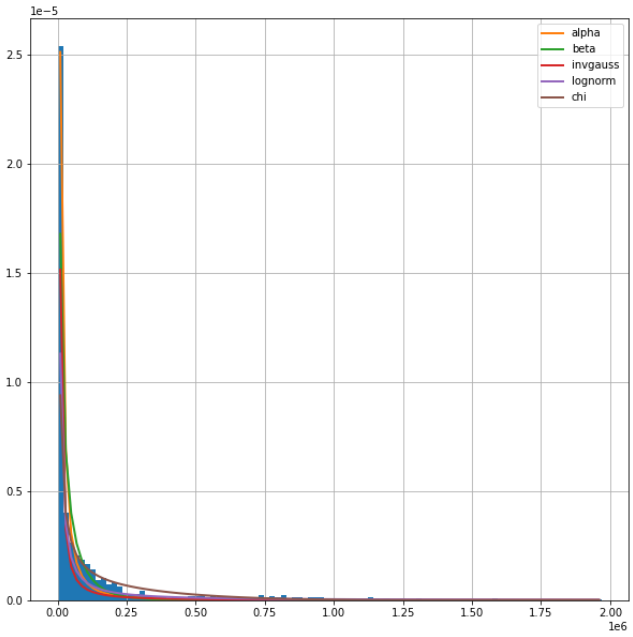
	sumsquare_error	aic	bic	kl_div
invgamma	4.282537e-12	3300.505365	-201771.391345	inf
invgauss	5.003453e-12	3329.988562	-200870.255398	inf
lognorm	2.921515e-11	3265.434897	-190649.929583	inf
t	3.470511e-11	3455.344580	-189652.549825	inf
beta	6.400124e-11	4262.308559	-186099.091606	inf



D'après ces résultats qui précèdent, les lois qui modélisent le plus la distribution de production de « Rice » est la loi inverse-gamma.

- Pour la distribution des productions de « Wheat » :

	sumsquare_error	aic	bic	kl_div
alpha	1.341679e-11	3566.630835	-84968.343095	inf
beta	8.652535e-11	4988.277120	-80144.090427	inf
invgauss	1.145422e-10	3493.566860	-79427.120829	inf
lognorm	2.032242e-10	3315.911967	-77945.541801	inf
chi	2.575413e-10	3579.676050	-77333.468153	inf



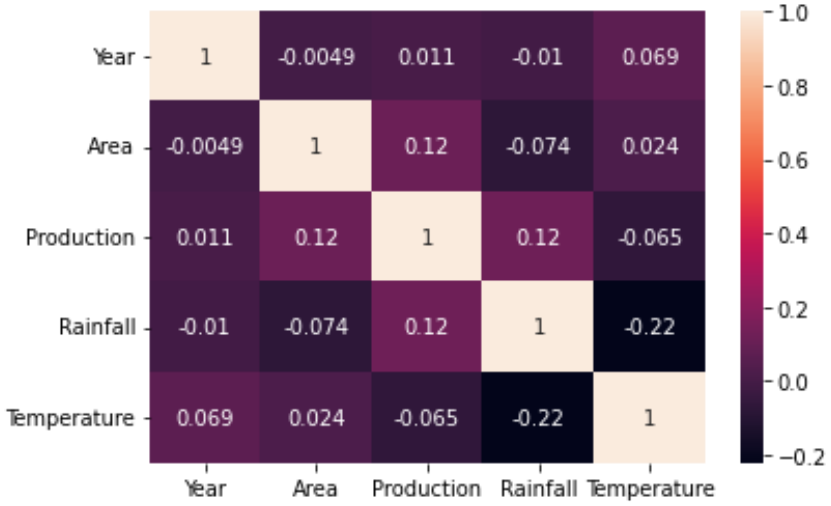
D'après ces résultats qui précèdent, les lois qui modélisent le plus la distribution de production de « Wheat » est alpha.

2. *Etude de l'indépendance des variables :*

Avant d'entamer la partie des modèles de prédiction, il est intéressant de voir les relations de dépendances entre les variables afin de préciser celles qui seront pertinents pour nos modèles et ceux qui ne le sont pas et qui peuvent être éliminés.

- ***Dépendance entre variables quantitatives :***

Pour visualiser les dépendances entre les différentes variables quantitatives, on utilise la matrice de corrélation.



On remarque que la variable « Température » est très faiblement corrélée avec la variable dépendante.

- ***Dépendance entre « Production » et variable qualitatives :***

Pour étudier la dépendance entre la variable dépendante et les différentes variables qualitatives, on utilise le test **Anova.**

Le test **Anova** permet de vérifier si la moyenne d'une variable numérique diffère selon les niveaux d'une variable catégorielle. Elle répond essentiellement à la question suivante : les moyennes des groupes diffèrent-elles les unes des autres ? En Python, on peut exécuter ce test en utilisant la bibliothèque **statsmodel.**

Si le modèle donne une valeur P (Prob F-statistique) inférieure au seuil de signification habituel de 0.05, nous concluons donc qu'il existe une relation significative entre les deux variables.

- District et Production :

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
anova = smf.ols(formula='Production ~ C(District)', data=data1).fit()
print(anova.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             Production   R-squared:                       0.187
Model:                            OLS   Adj. R-squared:                  0.169
Method:                 Least Squares   F-statistic:                     10.30
Date:                Tue, 22 Feb 2022   Prob (F-statistic):               0.00
Time:                        14:33:43   Log-Likelihood:            -2.2366e+05
No. Observations:               11513   AIC:                         4.478e+05
Df Residuals:                   11261   BIC:                         4.497e+05
Df Model:                         251
Covariance Type:            nonrobust
```

Le test donne une valeur P (Prob F-statistique) de 0.0<0.05, donc il y a une dépendance significative entre District et Production. Ainsi, on ne peut pas éliminer cette variable même s'il a un grand nombre de valeurs uniques (253) comme il est déjà cité auparavant.

- State et Production :

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
anova = smf.ols(formula='Production ~ C(State)', data=data1).fit()
print(anova.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             Production   R-squared:                       0.123
Model:                            OLS   Adj. R-squared:                  0.122
Method:                 Least Squares   F-statistic:                     146.5
Date:                Mon, 21 Mar 2022   Prob (F-statistic):          4.57e-317
Time:                        00:57:32   Log-Likelihood:            -2.2409e+05
No. Observations:               11513   AIC:                         4.482e+05
Df Residuals:                   11501   BIC:                         4.483e+05
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
```

De même, « State » et « Production » sont dépendants.

- Season et Production :

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
anova = smf.ols(formula='Production ~ C(Season)', data=data1).fit()
print(anova.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             Production   R-squared:                       0.055
Model:                            OLS   Adj. R-squared:                  0.055
Method:                 Least Squares   F-statistic:                     168.8
Date:                Mon, 21 Mar 2022   Prob (F-statistic):          1.17e-140
Time:                        01:02:25   Log-Likelihood:            -2.2452e+05
No. Observations:               11513   AIC:                         4.490e+05
Df Residuals:                   11508   BIC:                         4.491e+05
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
```

De même pour « Season » et « Production ».

III. Modélisation :

1. *Préparation à la modélisation :*

- ***Factorisation des variables qualitatives***

Comme les algorithmes d'apprentissage ne peuvent pas traiter les variables qualitatives, il est nécessaire de les convertir en variables numériques. Pour ce faire, on utilise la méthode qui, pour chaque variable qualitative, crée autant de variables binaires que de catégories présentes dans la variable.

```
categorical_features=['State', 'District', 'Crop', 'Season']
data1 = pd.get_dummies(data1,columns =categorical_features)
data1
```

	Year	Area	Production	Rainfall	Temperature	State_Andaman and Nicobar Islands	State_Arunachal Pradesh	State_E
0	1997	17876.0	21397.0	1303.7	31.77	0	0	
1	1997	38.0	59.0	1303.7	31.77	0	0	
2	1997	7812.0	11698.0	1303.7	31.77	0	0	
3	1997	1215.0	1867.0	1303.7	31.77	0	0	
4	1997	682.0	733.0	1303.7	31.77	0	0	
...	...	...	...	...	...	...	...	
74955	2015	91000.0	98000.0	1210.1	29.90	0	0	
74956	2015	1.0	52.0	1210.1	29.90	0	0	
74964	2015	86000.0	209000.0	1210.1	29.90	0	0	
74968	2015	17.0	1173.9	1210.1	29.90	0	0	
74974	2015	114000.0	156000.0	1210.1	29.90	0	0	

11513 rows × 278 columns

- ***Choix des variables***

En se basant sur les études exploratoire et statistique de notre data, plus particulièrement l'analyse univariée et l'étude de corrélation, on décide de ne pas tenir en compte la variable « Température » vu que sa variance et son coefficient de corrélation avec la variable dépendante sont très faibles.

- ***Répartition en Train et Test***

Avant de commencer à mettre en place des modèles d'apprentissage pour la prédiction de la Production, nous procédons à la création d'une base d'apprentissage (Train set) et une de test (Test set). En fait, nous avons opté

pour un découpage aléatoire avec 70% des données dans la base d'apprentissage servant à entrainer les modèles de prédiction et 30% dans la base de test.

```
# # we start by splitting our dataset to independent and target variable
x = data1.drop(columns=['Production', 'Temperature'],axis=1)
y=data1['Production']
# x = x.drop(labels=correlated_features, axis=1)
from sklearn.model_selection import train_test_split
import random
random.seed(90)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)
```

2. *Prédictions par algorithmes d'apprentissage :*

Nous évaluons nos modèles en se basant sur deux métriques qui sont : le **coefficient de détermination (R-squared)** et la **racine de l'erreur quadratique moyenne (RMSE)**.

Théoriquement, ils se calculent à l'aide des formules suivantes :

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

Où n est le nombre de mesures, y_i est la valeur de la ième observation du jeu de données de validation, $\bar{y}$ est la moyenne des valeurs du jeu de données de validation et $\hat{y}$ est la valeur prédite pour la ième observation.

- ***Arbres de décision et de régression :***

L'arbre de décision et de régression construit un modèle de régression sous la forme d'un arbre. Il décompose un ensemble de données en sous-ensembles de plus en plus petits tandis qu'un arbre de décision associé est développé progressivement. Le résultat final est un arbre avec des nœuds feuilles correspondant aux valeurs prédites et des nœuds de décision qui construisent les différentes combinaisons de variables d'entrée qui aboutissent aux résultats.

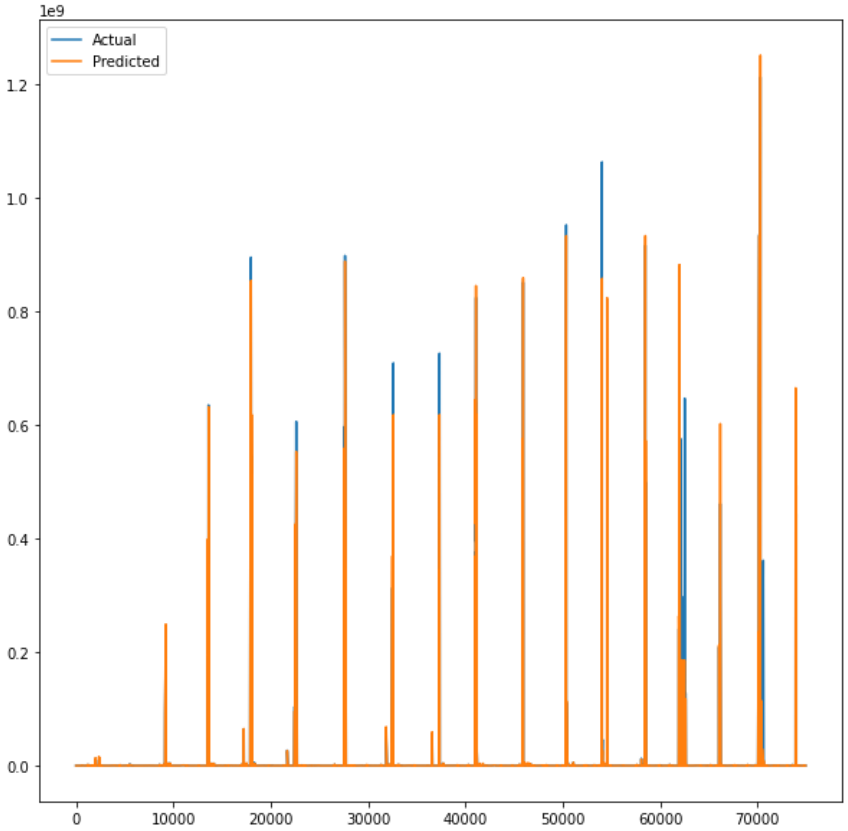
```
from sklearn.tree import  DecisionTreeRegressor
dtr = DecisionTreeRegressor(random_state=0)
rt = dtr.fit(x_train, y_train)
```

Métriques d'évaluation :

```
R_squared of CART on train set: 1.00
Train RMSE: 0.0
R_squared of CART on test set: 0.92
Test RMSE: 21830676.204421364
```

Nous obtenons un R^2 le test de 0.93, qui est proche de 1, ainsi le modèle donne de bonnes prédictions.

Comparaison des valeurs prédites et réelles :



- ***Forets aléatoires de régression :***

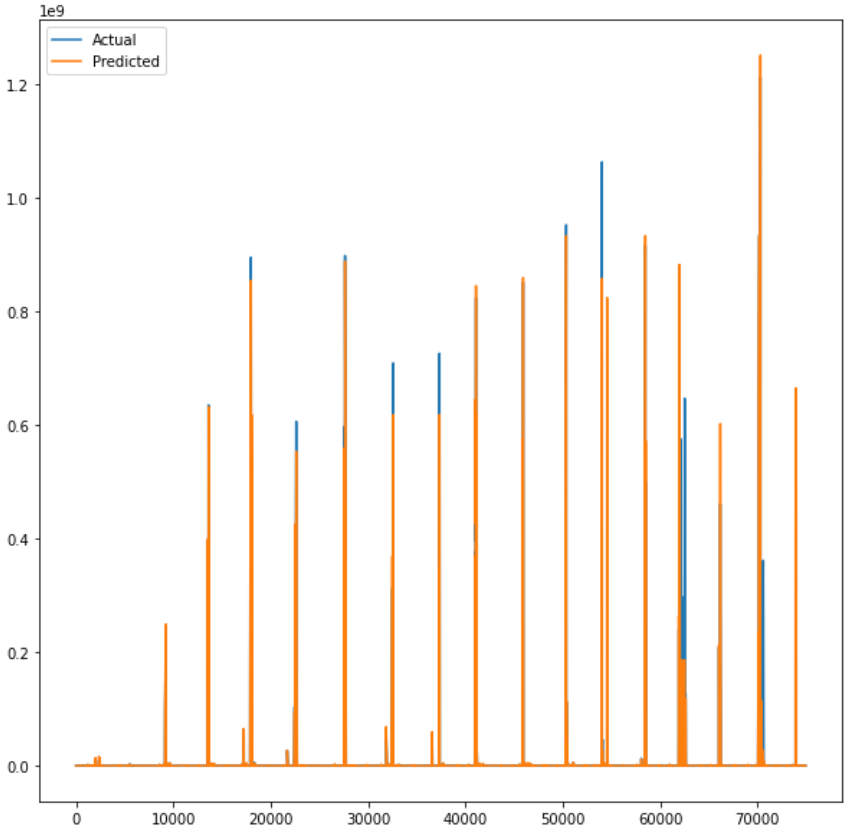
L'algorithme de forêt aléatoire fusionne la sortie de plusieurs arbres de décision pour générer la sortie finale.

```
from sklearn.ensemble import RandomForestRegressor
Model = RandomForestRegressor(n_estimators = 1000,random_state=0)
Model.fit(x_train,y_train)
```

Métriques d'évaluation :

```
Train R_squared: 0.99
Train RMSE: 7763175.749703884
Test R_squared: 0.92
Test RMSE: 20567835.90354847
```

Comparaison des valeurs prédites et réelles :



- ***XGBoost Regressor :***

« eXtreme Gradient Boosting » est une implémentation efficace de l'algorithme d'arbres à gradient boosté. Le gradient boosting est un algorithme qui tente à prédire avec précision une variable cible en combinant les estimations d'un ensemble de modèles plus simples et plus faibles.

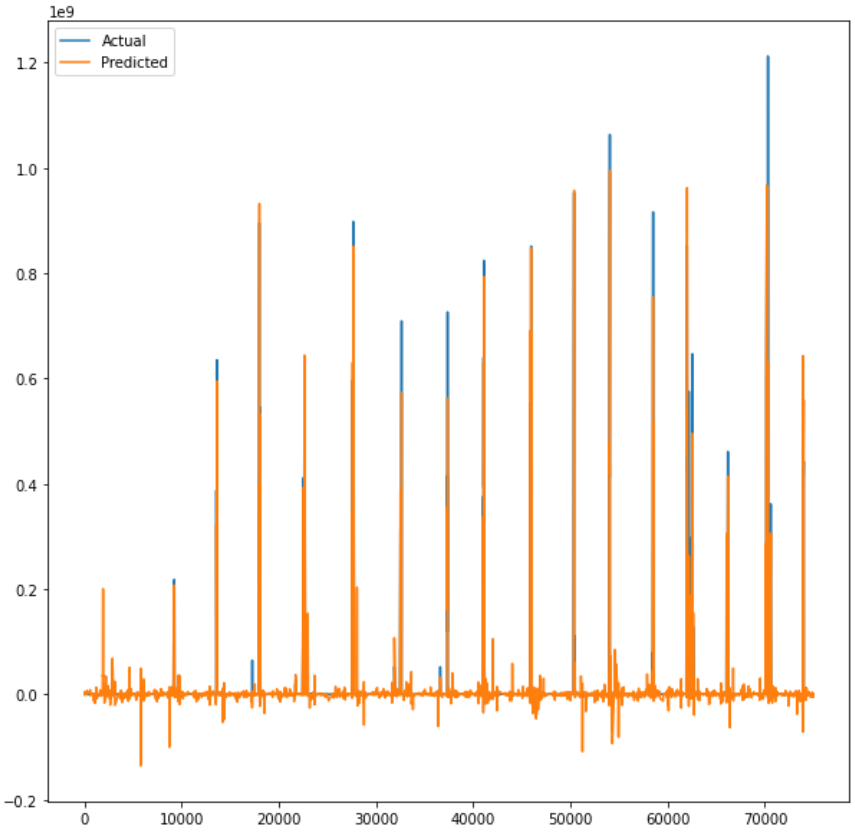
On effectue un ajustement des paramètres (l'hyper parameters tuning) afin de trouver les meilleurs paramètres du modèle :

```
from  sklearn.model_selection import RandomizedSearchCV
import xgboost
xgb = xgboost.XGBRegressor()
params={'learning_rate':[0.9, 0.8, 0.7, 0.6],'max_depth':[30, None], 'gamma':[1],
        'n_estimators':[400],'colsample_bytree':[0.8], 'min_child_weight':[15, 8, None],
        'subsample':[0.8],'objective':['reg:squarederror'], 'seed':[0]}
gridxgb = RandomizedSearchCV(xgb,
                         params,
                         cv=5,
                         n_jobs=-1,
                         scoring='neg_root_mean_squared_error'
                         )
gridxgb=gridxgb.fit(x_train,y_train)
y_pred1=gridxgb.predict(x_train)
y_pred=gridxgb.predict(x_test)
```

Métriques d'évaluation :

```
Train R_squared : 0.9999872744310164
Test R_squared: 0.9417988890269795
Train RMSE: 259216.3659409811
Test RMSE: 18081070.5492123
```

Comparaison des valeurs prédites et réelles :



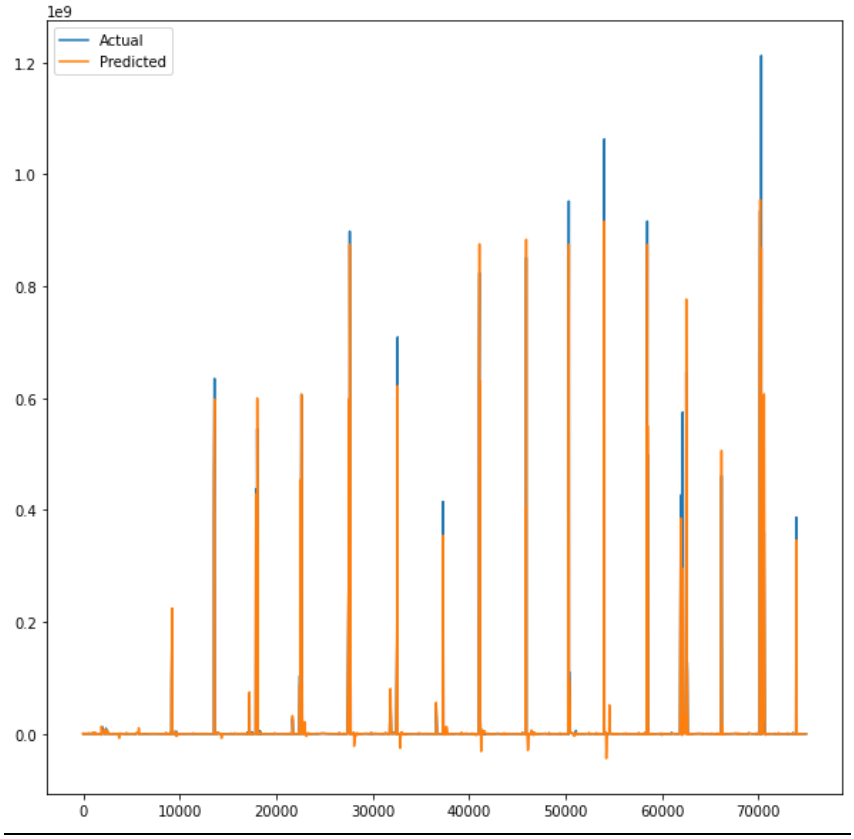
- ***Catboost :***

Le Catboost est un algorithme d'apprentissage automatique d'arbres de décision boosté développé par Yandex. Il fonctionne de la même manière que d'autres algorithmes boostés par gradient tels que XGBoost mais, entre autres, à un niveau de précision plus élevé sans paramètres de réglage.

Métriques d'évaluation :

```
R_squared on train set: 0.9944731580826233    Train RMSE: 5298825.147171965
R_squared on test set: 0.9735291021479913     Test RMSE: 13200705.261895072
```

Comparaison des valeurs prédites et réelles :



3. *Comparaison des modèles :*

En comparant les cinq modèles, on conclut que le Catboost est le plus performant puisqu'il a la plus petite valeur de RMSE et la valeur de R^2 sur le test le plus proche à 1.

	R^2	RMSE
Arbres de décision et de régression	0.92	20 042 432.98
Forets aléatoires de régression	0.92	20 619 358.13
XGBoost Regressor	0.94	18 541 205.68
Catboost	0.97	13 200 705.26

Cependant, les deux modèles XGBoost et Catboost, prédisent parfois des valeurs négatives qui sont enfaite impossible pour notre cas puisque les productions ne peuvent pas être de valeurs négatives.

Bien que nous n'ayons pas observé de valeurs négatives dans notre phase de Training, si le modèle est basé sur des fonctions, de nouvelles valeurs d'entrée pourraient conduire à une prédiction négative par extrapolation. Les seules façons de ne JAMAIS obtenir de valeurs prédites négatives seraient soit d'utiliser un modèle qui intrinsèquement ne peut pas prédire des valeurs négatives (comme certains types de fonctions de lien dans les modèles GLM tels

que Poisson ou logistique), soit d'utiliser un modèle qui n'extrapole pas (par ex. , certains types de modèles d'arbres). Sinon, nous ne pouvons pas garantir que le modèle ne verra pas de nouvelles données d'entrée et ne générera pas une prédiction en dehors de la plage prédite précédemment (qui pourrait être négative).

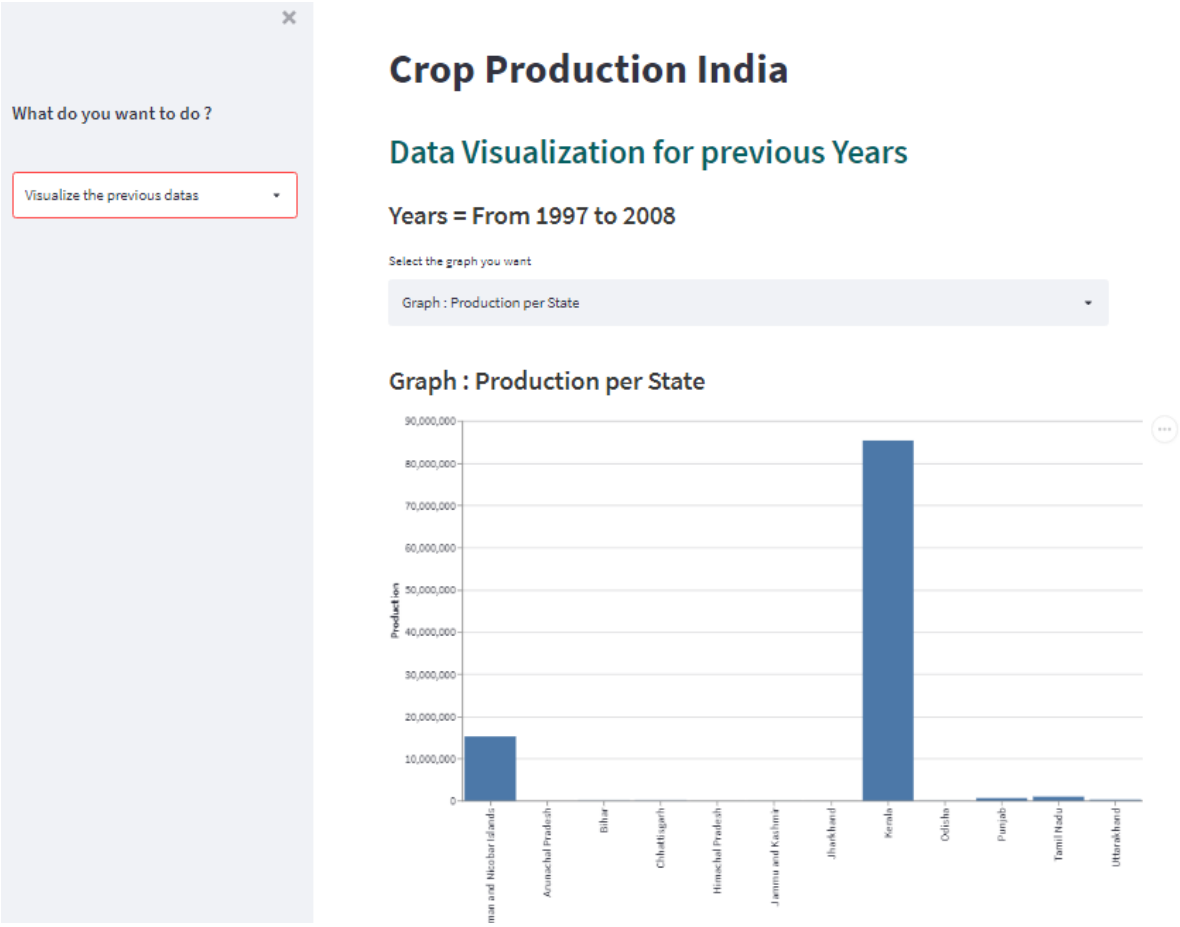
Un possible « détour » qu'on peut utiliser face à ce problème est de donner manuellement la valeur de 0 à toutes les prédictions négatives.

IV. Interface graphique :

Pour mieux satisfaire le client et facilité l'utilisation de notre modèle ; on a réalisé une interface graphique qui permet la visualisation de la data et aussi la prédiction en fonction de la date, de la région, de l'état et de la culture. Alors ; notre interface et devisée en deux parties ; la première est la visualisation de data et la deuxième est la prédiction.

- **La visualisation de la data :**

Cette partie permet au client de la visualisation de la data entre les années en trois graphes Production par State, Production par Crop et la productivité de chaque State. Ainsi que ; la visualisation est entre l'année 1997 et 2008, en considérant que la data dans ce intervalle est la data de trainning.



- **La prédiction :**

Dans cette partie ; sur l'application mobile on permet du client de visualisation de la data prédit. Ainsi ; on visualise la production en fonction des années et en fonction de la District et aussi l'affichage de la data toute entier. Alors ; pour cela ; en prenant la data de test entre les années 2008 et 2015 et faisant la prédiction par notre meilleur modèle qu'on a utlisé

What do you want to do ?

Make a prediction

Input parameters for the prediction

Select the start date

2009

Select the end date

2014

Select the State

Kerala

Select the Crop

Sugarcane

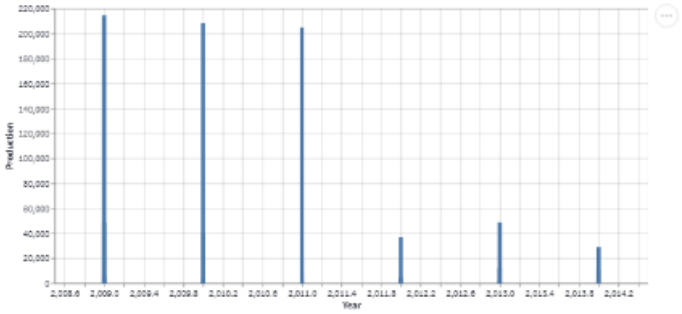
Predict

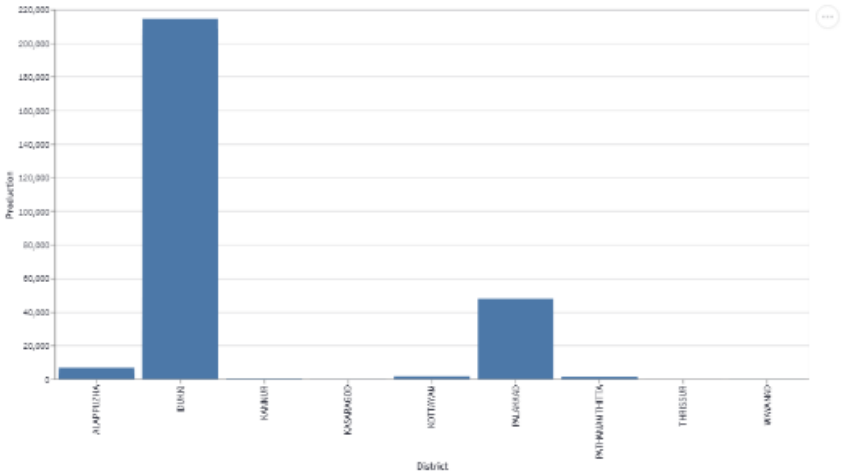
Crop Production India

Crop production: Kerala Crop: Sugarcane between 2009 and 2014

	State	District	Year	Season	Crop	Area	Production	Rainfall	Temperature
53965	Kerala	ALAPPUZHA	2009	Whole Year	Sugarcane	66.0000	5,191.9420	2,810.6000	30.3000
53986	Kerala	IDUKKI	2009	Whole Year	Sugarcane	2,201.0000	214,613.9310	2,810.6000	30.3000
53993	Kerala	KANNUR	2009	Whole Year	Sugarcane	1.0000	139.7777	2,810.6000	30.3000
54003	Kerala	KASARAGOD	2009	Whole Year	Sugarcane	1.0000	62.8172	2,810.6000	30.3000
54024	Kerala	KOTTAYAM	2009	Whole Year	Sugarcane	31.0000	1,448.2724	2,810.6000	30.3000
54055	Kerala	PALAKKAD	2009	Whole Year	Sugarcane	647.0000	47,809.0021	2,810.6000	30.3000
54065	Kerala	PATHANAMTHITTA	2009	Whole Year	Sugarcane	9.0000	732.7690	2,810.6000	30.3000
58425	Kerala	ALAPPUZHA	2010	Whole Year	Sugarcane	54.2500	3,519.2725	3,131.8000	30.1300
58444	Kerala	IDUKKI	2010	Whole Year	Sugarcane	2,173.5300	208,431.2200	3,131.8000	30.1300
58450	Kerala	KANNUR	2010	Whole Year	Sugarcane	1.2400	125.0161	3,131.8000	30.1300

Visualize the predictions with graphs





Conclusion

Dans le cadre du cours statistiques en 2ème année cycle ingénieur, il nous a été confié, en tant que groupe, de mener une étude sous forme de projet à propos de la prédiction de la récolte de différentes cultures en Inde en fonction de la géographie, la saison, et la surface cultivée. Pour ce faire, des tests statistiques, des modèles de machine learning, et des visualisations de données ont été réalisés.

La démarche suivie se compose principalement de trois parties majeures : l'étude descriptive, l'étude statistique et la modélisation. En effet, on a débuté par faire un diagnostic de notre dataset afin de chercher les éventuelles anomalies (valeurs manquantes et aberrantes) où le seul problème observé était l'irrégularité de saisie dans quelques valeurs. Par la suite, on a réalisé une description détaillée de toutes les variables qualitatives ainsi qu'une analyse descriptive univariée des variables quantitatives et une analyse multivariée des données.

La deuxième grande partie concerne l'étude statistique, où on cherche la loi de distribution usuelle la plus proche que suit notre variable dépendante «Production». Pour ce faire, on a recouru à la méthode automatique en utilisant La bibliothèque « fitter » dans pyhton qui fournit des méthodes simples permettant d'identifier la distribution qui modélise le mieux une distribution de données.

Ensuite, on a achevé cette deuxième grande partie en s'intéressant à l'étude l'indépendance des variables en étudiant le coefficient de corrélation qui quantifie la force de la relation linéaire entre la variable dépendante et les autres variables indépendantes, ainsi que le test Anova qui permet de vérifier si la moyenne d'une variable numérique diffère selon les niveaux d'une variable catégorielle.

Dans la troisième partie, on a achevé notre étude par l'application de plusieurs modèles statistiques pour pouvoir prédire la production, à savoir « Arbres de décision et de régression », « Forets aléatoires de régression » , « XGBoost Regressor » et le « Catboost », et la comparaison entre ces modèles.

Finalement, on a réalisé une interface qui permet la visualisation et la prédiction de la data dans un intervalle de temps déterminé par l'utilisateur.