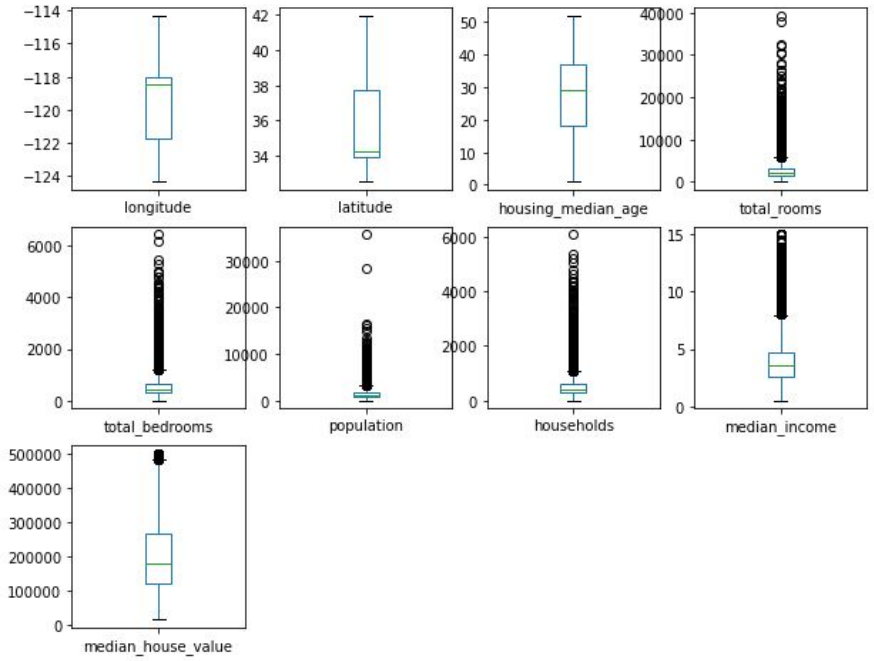
Prédictions de prix immobiliers en Californie

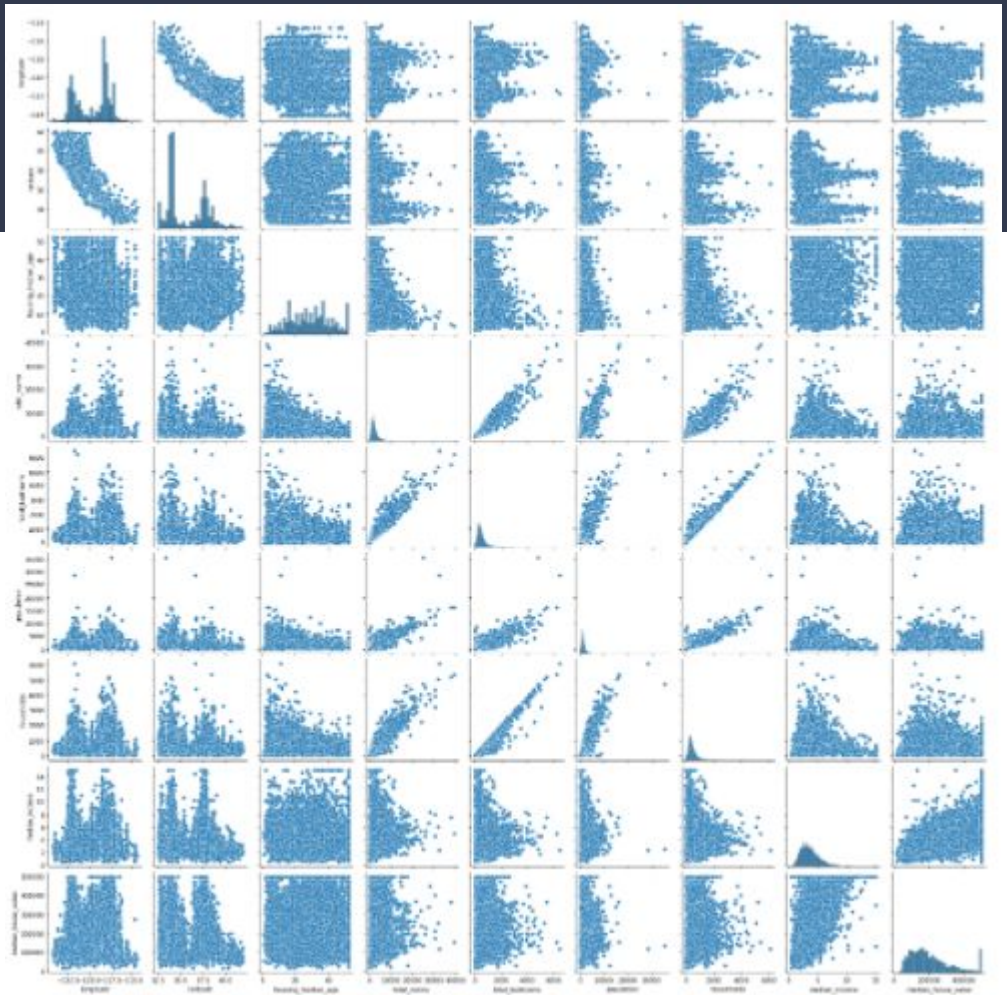
Objectifs

- **Estimer la valeur du prix médian des maisons par district / bloc**
- Variables fournies : longitude, latitude, âge médian du district, nombre total de pièces dans un bloc, nombre total de chambres dans un bloc, nombre total de personnes résidant dans un bloc, nombre total de ménages pour un bloc, revenu médian des ménages dans un bloc proximité du bloc par rapport à la mer

- Exploration et nettoyage du jeu de données
- Construction d’un modèle de données et optimisation des paramètres
- Inférence
- Prédictions

EDA : exploration des données





Nettoyage des données & Preprocessing

- Suppression de la colonne “Unamed_0”
- Suppression des enregistrements avec des données manquantes

- Normalisation des données

- Encodage de ocean_proximity

Sélection des features

- feature permutation

⇒ suppression des variables <1H_OCEAN et ISLAND

- VIF

Régression Linéaire

Entrée [302]:
```
# Choose your model
lr = LinearRegression()
# 5-Fold Cross validate model
cv_results = cross_validate(lr, X, y, cv=5)
# obtain the mean of scores
cv_results['test_score'].mean()
```

Out[302]:
```
0.6114982101008223
```

Entrée [303]:
```
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
print(f"MSE : {mean_squared_error(y_test, y_pred)}")
print(f"RMSE : {np.sqrt(mean_squared_error(y_test, y_pred))}")
print(f"MAE : {mean_absolute_error(y_test, y_pred)}")
```

```
MSE : 5071439846.444097
RMSE : 71214.04248070809
MAE : 52308.24522306676
```

Inférence

OLS Regression Results

Dep. Variable:	y_train	R-squared:	0.649
Model:	OLS	Adj. R-squared:	0.649
Method:	Least Squares	F-statistic:	1922.
Date:	Fri, 24 Sep 2021	Prob (F-statistic):	0.00
Time:	11:10:56	Log-Likelihood:	-1.4345e+05
No. Observations:	11428	AIC:	2.869e+05
Df Residuals:	11416	BIC:	2.870e+05
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.611e+05	1.08e+04	24.282	0.000	2.4e+05	2.82e+05
X_train[0]	-2.698e+05	1.38e+04	-19.603	0.000	-2.97e+05	-2.43e+05
X_train[1]	-2.438e+05	1.27e+04	-19.204	0.000	-2.69e+05	-2.19e+05
X_train[2]	5.543e+04	2981.789	18.590	0.000	4.96e+04	6.13e+04
X_train[3]	-2.145e+05	4.01e+04	-5.352	0.000	-2.93e+05	-1.36e+05
X_train[4]	6.206e+05	5.99e+04	10.366	0.000	5.03e+05	7.38e+05
X_train[5]	-1.234e+06	4.33e+04	-28.476	0.000	-1.32e+06	-1.15e+06
X_train[6]	4.047e+05	6.13e+04	6.598	0.000	2.84e+05	5.25e+05
X_train[7]	5.643e+05	6562.645	85.991	0.000	5.51e+05	5.77e+05
X_train[8]	-2894.4364	2084.750	-1.388	0.165	-6980.905	1192.032
X_train[9]	-4.209e+04	2995.571	-14.050	0.000	-4.8e+04	-3.62e+04
X_train[10]	-6535.2593	2894.919	-2.257	0.024	-1.22e+04	-860.720

Omnibus:	2474.016	Durbin-Watson:	2.019
Prob(Omnibus):	0.000	Jarque-Bera (JB):	8209.411
Skew:	1.086	Prob(JB):	0.00
Kurtosis:	6.538	Cond. No.	190.

KNN Regression

```
model = KNeighborsRegressor()
# 5-Fold Cross validate model
cv_results = cross_validate(model, X, y, cv=5)
# obtain the mean of scores
cv_results['test_score'].mean()
```

```
0.7069975382669904
```

Obtention d’un meilleur score que le modèle de régression linéaire

Prédictions

- **notebook_predict**

Récupération de la base de données sous forme de dataframe :

```
df = pd.read_csv(r'C:\Users\Admin\Documents\marianneSimplon\simplon\immo_SiliconValley_marianneD\data\traindata_ori.csv',delimit
```

- Remplacer le chemin d'accès (en rouge)
- Transformation des données
- Récupération des prédictions sous forme de Datarame :

```
predictions = my_model.predict(X)
predictions = pd.DataFrame(predictions)
predictions
```

	0
0	109160.0
1	207000.0
2	158100.0
3	436500.8
4	143920.0
...	...
16331	468640.6
16332	390240.0
16333	74420.0
16334	114760.0
16335	143700.0

16336 rows × 1 columns

Pistes d’amélioration du modèle

- regularization
- model tuning
- ...