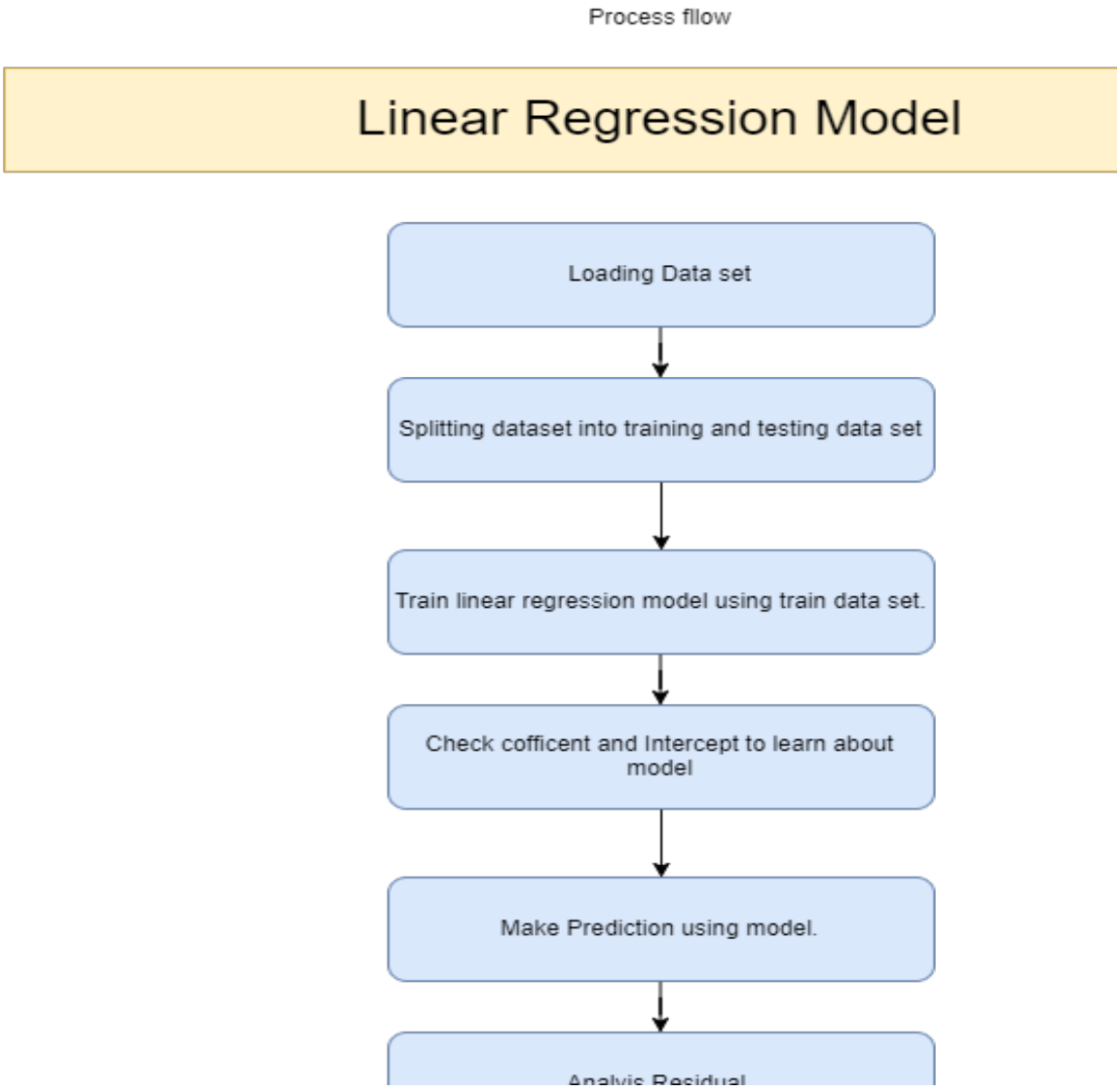
Linear Regression Model

Regression is a method of modeling a target value based on independent predictors. This method is mostly used for forecasting and finding out cause and effect relationship between variables. Regression techniques mostly differ based on the number of independent variables and the type of relationship between the independent and dependent variables.

Process Flow



Codes:-

- **To import following libraries.**

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

- To read CSV file using pandas and load dataset into df

```
df = pd.read_csv('USA_Housing.csv')
```

- To view, info, describe and to see column of data set

```
df.head()
df.info()
df.describe()
df.columns
```

- To import machine learning library with various models.

```
from sklearn.model_selection import train_test_split
```

- To Create variable X and y for training and testing sets. [X = Column names] [y = Column against which we have to plot]

```
X = df[['Avg. Area Income','Avg. Area House Age','Avg. Area Number of Rooms','Avg. Area Number of Bedrooms', 'Area Population']]
y = df['Price']
```

- To Create Test and Training data sets.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=101)
```

- Import Linear regression library from sklearn

```
from sklearn.linear_model import LinearRegression
```

- Create linear regression object.

```
lm = LinearRegression()
```

- To Train model.

```
lm.fit(X_train,y_train)
```

- To view Intercept and Coefficient

```
print(lm.intercept_)
lm.coef_
```

- To put coefficient in data frame.

```
cdf = pd.DataFrame(lm.coef_,X.columns,columns=['Coeff'])
```

Predictions

- To create prediction

```
predictions = lm.predict(X_test)
```

- To Create Scatter plot

```
plt.scatter(y_test,predictions)
```

- To plot histogram

```
sns.distplot((y_test-predictions))
```

To Test Predictions

Here are three common evaluation metrics for regression problems:

Mean Absolute Error (MAE) is the mean of the absolute value of the errors:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Mean Squared Error (MSE) is the mean of the squared errors:

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

Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors:

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

Comparing these metrics:

- **MAE** is the easiest to understand, because it's the average error.
- **MSE** is more popular than MAE, because MSE "punishes" larger errors, which tends to be useful in the real world.
- **RMSE** is even more popular than MSE, because RMSE is interpretable in the "y" units.

All of these are **loss functions**, because we want to minimize them.

Code to use metrics

```
from sklearn import metrics

metrics.mean_absolute_error(y_test,predictions)

metrics.mean_squared_error(y_test,predictions)

np.sqrt(metrics.mean_squared_error(y_test,predictions))
```