Imputing using fancyimpute

DEALING WITH MISSING DATA IN PYTHON

Suraj Donthi
Deep Learning & Computer Vision Consultant

fancyimpute package

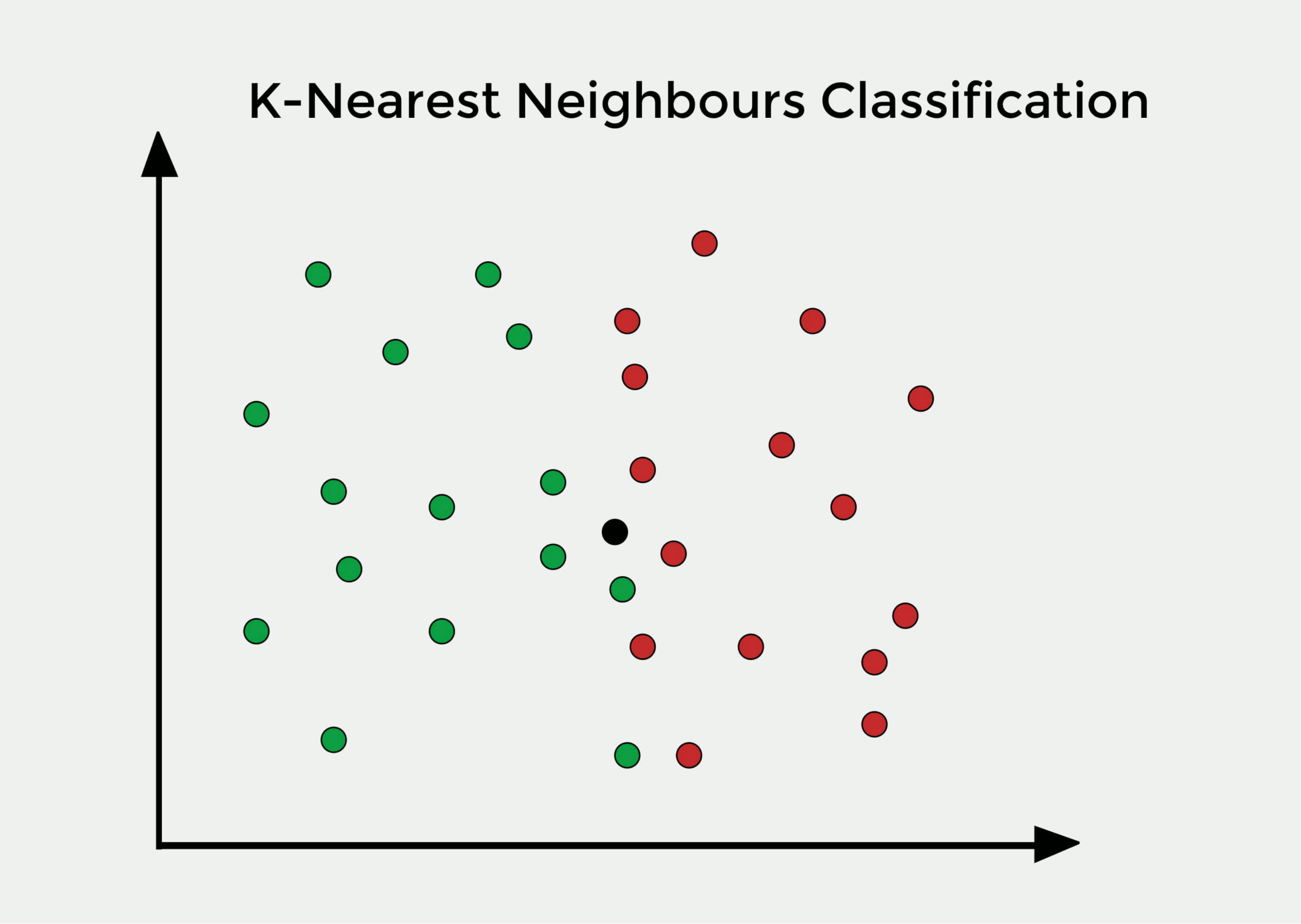
- Package contains advanced techniques
- Uses machine learning algorithms to impute missing values
- Uses other columns to predict the missing values and impute them

Fancyimpute imputation techniques

- KNN or K-Nearest Neighbor
- MICE or Multiple Imputation by Chained Equations

K-Nearest Neighbor Imputation

- Select K nearest or similar data points using all the non-missing features
- Take average of the selected data points to fill in the missing feature



K-Nearest Neighbor Imputation

```
from fancyimpute import KNN
diabetes_knn = diabetes.copy(deep=True)
knn_imputer = KNN()
diabetes_knn.iloc[:, :] = knn_imputer.fit_transform(diabetes_knn)
```

Multiple Imputations by Chained Equations (MICE)

- Perform multiple regressions over random sample of the data
- Take average of the multiple regression values
- Impute the missing feature value for the data point

Multiple Imputations by Chained Equations(MICE)

```
from fancyimpute import IterativeImputer
diabetes_MICE = diabetes.copy(deep=True)
MICE_imputer = IterativeImputer()
diabetes_MICE.iloc[:, :] = MICE_imputer.fit_transform(diabetes_MICE)
```

Summary

- Using Machine Learning techniques to impute missing values
- KNN finds most similar points for imputing
- MICE performs multiple regression for imputing
- MICE is a very robust model for imputation

Let's practice!

DEALING WITH MISSING DATA IN PYTHON

Imputing categorical values

DEALING WITH MISSING DATA IN PYTHON

Suraj Donthi
Deep Learning & Computer Vision Consultant

Complexity with categorical values

- Most categorical values are strings
- Cannot perform operations on strings
- Necessity to convert/encode strings to numeric values and impute

Conversion techniques

ONE-HOT ENCODER

Color	Color_Red	Color_Green	Color_Blue
Red	1	0	0
Green	0	1	0
Blue	0	0	1
Red	1	0	0
Blue	0	0	1
Blue	0	0	1

ORDINAL ENCODER

Color	Value
Red	0
Green	1
Blue	2
Red	0
Blue	2
Blue	2

Imputation techniques

- Fill with most frequent category
- Impute using statistical models like KNN

Users profile data

```
users = pd.read_csv('userprofile.csv')
users.head()
```

	smoker	drink_level	dress_preference	ambience	hijos	activity	budget
0	False	abstemious	informal	family	independent	student	medium
1	False	abstemious	informal	family	independent	student	low
2	False	social drinker	formal	family	independent	student	low
3	False	abstemious	informal	family	independent	professional	medium
4	False	abstemious	no preference	family	independent	student	medium

Ordinal Encoding

```python
from sklearn.preprocessing import OrdinalEncoder

# Create Ordinal Encoder
ambience_ord_enc = OrdinalEncoder()

# Select non-null values in ambience
ambience = users['ambience']
ambience_not_null = ambience[ambience.notnull()]
reshaped_vals = ambience_not_null.values.reshape(-1, 1)
# Encode the non-null values of ambience
encoded_vals = ambience_ord_enc.fit_transform(reshaped_vals)
# Replace the ambience column with ordinal values
users.loc[ambience.notnull(), 'ambience'] = np.squeeze(encoded_vals)
```

Ordinal Encoding

```
# Create dictionary for Ordinal encoders
ordinal_enc_dict = {}

# Loop over columns to encode
for col_name in users:
    # Create ordinal encoder for the column
    ordinal_enc_dict[col_name] = OrdinalEncoder()
    col = users[col_name]

    # Select the non-null values in the column
    col_not_null = col[col.notnull()]
    reshaped_vals = col_not_null.values.reshape(-1, 1)

    # Encode the non-null values of the column
    encoded_vals = ordinal_enc_dict[col_name].fit_transform(reshaped_vals)

    # Replace the values in the column with ordinal values
    users.loc[col.notnull(), col_name] = np.squeeze(encoded_vals)
```

Imputing with KNN

```
users_KNN_imputed = users.copy(deep=True)

# Create KNN imputer
KNN_imputer = KNN()

users_KNN_imputed.iloc[:, :] = np.round(KNN_imputer.fit_transform(users))

for col_name in users_KNN_imputed:
    # Reshape the values to 2-dimensions to
    # avoid errors while storing in the DataFrame
    reshaped = users_KNN_imputed[col_name].values.reshape(-1, 1)
    users_KNN_imputed[col_name] = \
    ordinal_enc_dict[col_name].inverse_transform(reshaped)
```

Summary

Steps to impute categorical values

- Convert non-missing categorical columns to ordinal values
- Impute the missing values in the ordinal DataFrame
- Convert back from ordinal values to categorical values

Let's practice!

DEALING WITH MISSING DATA IN PYTHON

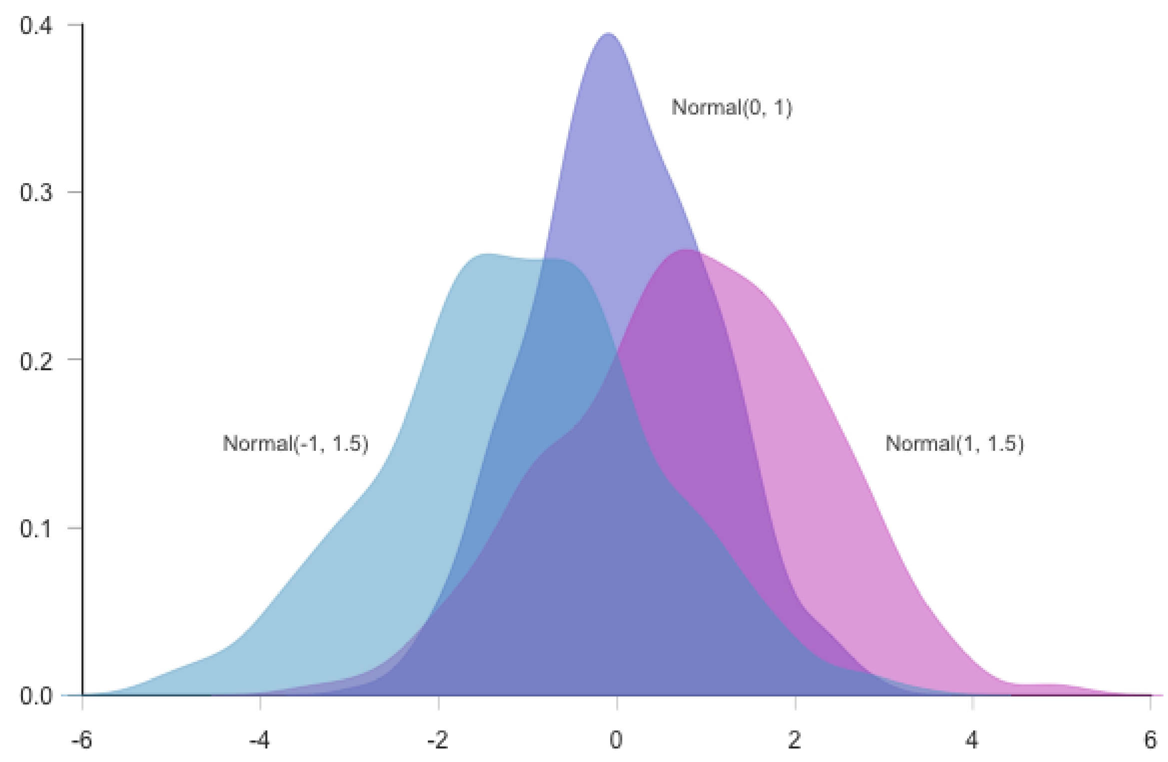
Evaluation of different imputation techniques

DEALING WITH MISSING DATA IN PYTHON

Suraj Donthi
Deep Learning & Computer Vision Consultant

Evaluation techniques

- Imputations are used to improve model performance.
- Imputation with maximum machine learning model performance is selected.
- Density plots explain the distribution in the data.
- A very good metric to check bias in the imputations.



Fit a linear model for statistical summary

```python
import statsmodels.api as sm

diabetes_cc = diabetes.dropna(how='any')
X = sm.add_constant(diabetes_cc.iloc[:, :-1])
y = diabetes_cc['Class']
lm = sm.OLS(y, X).fit()
```

```python
print(lm.summary())
```

```
Summary:                              OLS Regression Results
==============================================================================
Dep. Variable:                  Class   R-squared:                       0.346
Model:                            OLS   Adj. R-squared:                  0.332
Method:                 Least Squares   F-statistic:                     25.30
Date:                Wed, 10 Jul 2019   Prob (F-statistic):           2.65e-31
Time:                        15:03:19   Log-Likelihood:                -177.76
No. Observations:                 392   AIC:                             373.5
Df Residuals:                     383   BIC:                             409.3
Df Model:                           8
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
<hr />----------------------------------------------------------------------------------
const               -1.1027      0.144     -7.681      0.000      -1.385      -0.820
Pregnant             0.0130      0.008      1.549      0.122      -0.003       0.029
Glucose              0.0064      0.001      7.855      0.000       0.005       0.008
Diastolic_BP      5.465e-05      0.002      0.032      0.975      -0.003       0.003
Skin_Fold            0.0017      0.003      0.665      0.506      -0.003       0.007
Serum_Insulin       -0.0001      0.000     -0.603      0.547      -0.001       0.000
BMI                  0.0093      0.004      2.391      0.017       0.002       0.017
Diabetes_Pedigree    0.1572      0.058      2.708      0.007       0.043       0.271
Age                  0.0059      0.003      2.109      0.036       0.000       0.011
```

R-squared and Coefficients

```
lm.rsquared_adj
```

```
0.33210
```

```
lm.params
```

```
const                -1.102677
Pregnant              0.012953
Glucose               0.006409
Diastolic_BP          0.000055
Skin_Fold             0.001678
Serum_Insulin        -0.000123
BMI                   0.009325
Diabetes_Pedigree     0.157192
Age                   0.005878
dtype: float64
```

Fit linear model on different imputed DataFrames

```
# Mean Imputation
X = sm.add_constant(diabetes_mean_imputed.iloc[:, :-1])
y = diabetes['Class']
lm_mean = sm.OLS(y, X).fit()
# KNN Imputation
X = sm.add_constant(diabetes_knn_imputed.iloc[:, :-1])
lm_KNN = sm.OLS(y, X).fit()
# MICE Imputation
X = sm.add_constant(diabetes_mice_imputed.iloc[:, :-1])
lm_MICE = sm.OLS(y, X).fit()
```

Comparing R-squared of different imputations

```
print(pd.DataFrame({'Complete': lm.rsquared_adj,
                          'Mean Imp.': lm_mean.rsquared_adj,
                          'KNN Imp.': lm_KNN.rsquared_adj,
                          'MICE Imp.': lm_MICE.rsquared_adj},
                         index=['R_squared_adj']))
```

```
                    Complete   Mean Imp.    KNN Imp.   MICE Imp.
R_squared_adj       0.332108    0.313781    0.316543    0.317679
```

> **Note:** The metrics used here is for linear correlations only. You must use the metrics that are reflective of the data.

Comparing coefficients of different imputations

```python
print(pd.DataFrame({'Complete': lm.params,
                    'Mean Imp.': lm_mean.params,
                    'KNN Imp.': lm_KNN.params,
                    'MICE Imp.': lm_MICE.params}))
```

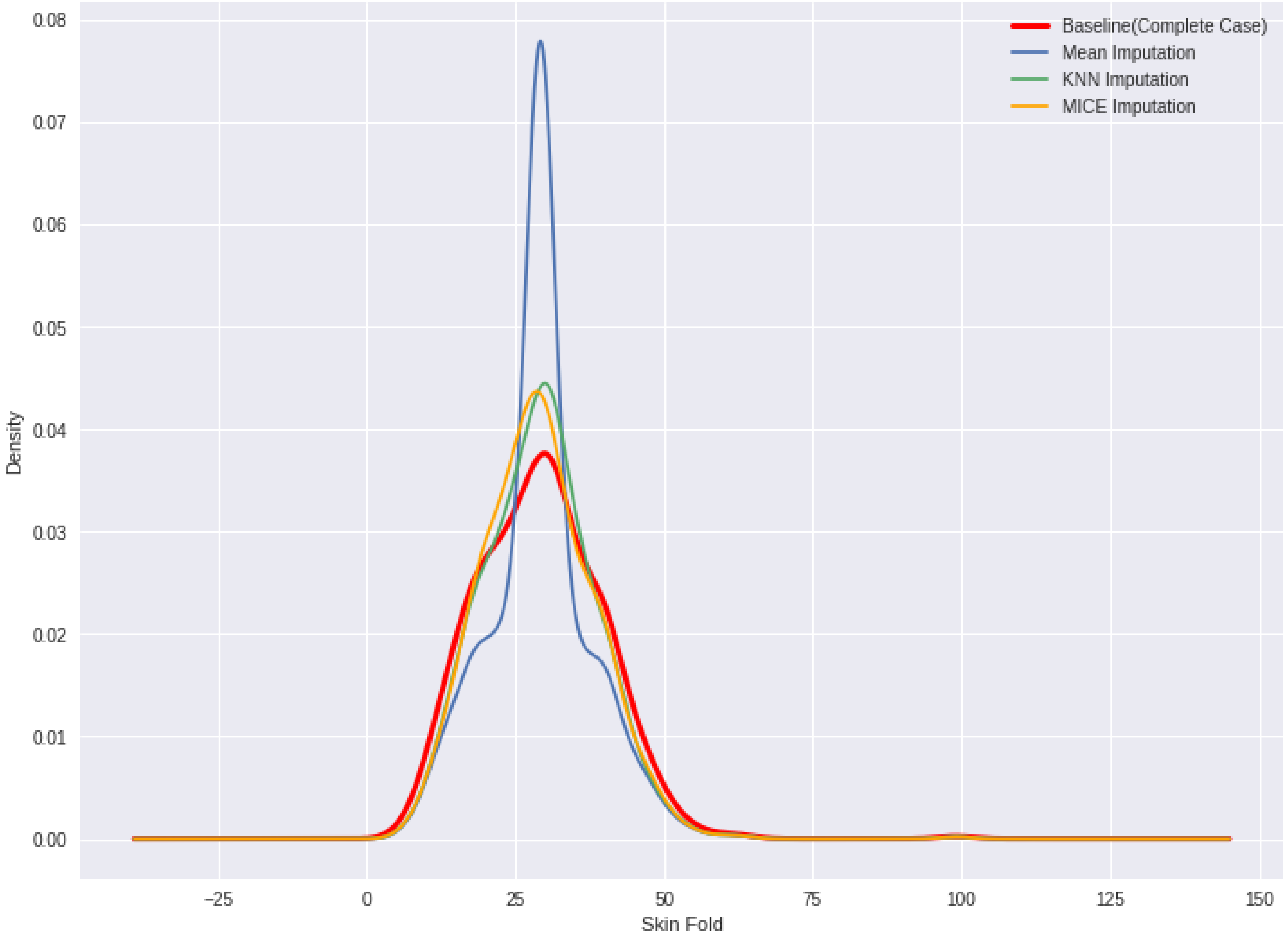
```
                   Complete  Mean Imp.  KNN Imp.  MICE Imp.
const             -1.102677  -1.024005  -1.028035  -1.050023
Pregnant           0.012953   0.020693   0.020047   0.020295
Glucose            0.006409   0.006467   0.006614   0.006871
Diastolic_BP       0.000055  -0.001137  -0.001196  -0.001317
Skin_Fold          0.001678   0.000193   0.001626   0.000807
Serum_Insulin     -0.000123  -0.000090  -0.000147  -0.000227
BMI                0.009325   0.014376   0.013239   0.014203
Diabetes_Pedigree  0.157192   0.129282   0.128038   0.129056
Age                0.005878   0.002092   0.002046   0.002097
```

Comparing density plots

```
diabetes_cc['Skin_Fold'].plot(kind='kde', c='red', linewidth=3)
diabetes_mean_imputed['Skin_Fold'].plot(kind='kde')
diabetes_knn_imputed['Skin_Fold'].plot(kind='kde')
diabetes_mice_imputed['Skin_Fold'].plot(kind='kde')

labels = ['Baseline (Complete Case)', 'Mean Imputation', 'KNN Imputation',
          'MICE Imputation']
plt.legend(labels)
plt.xlabel('Skin Fold')
```

Comparing density plots



Summary

- Applying linear model from the statsmodels package
- Comparing the coefficients and standard errors
- Comparing density plots

Let's practice!

DEALING WITH MISSING DATA IN PYTHON

Conclusion

DEALING WITH MISSING DATA IN PYTHON

Suraj Donthi
Deep Learning & Computer Vision Consultant

Chapter 1

- Null Value operations
- Detecting missing values
- Replacing missing values
- Analyzing amount of missingness

Chapter 2

- Types of missingness
  - MCAR
  - MAR
  - MNAR
- Correlations of missingness
  - Heatmaps
  - Dendrograms
- Visualize missingness across a variable
- Deleting missing values

Chapter 3

- Imputation techniques
- Treating time-series data
- Graphical comparison of imputed time-series data

Chapter 4

- Advanced imputation techniques
  - KNN
  - MICE
- Imputing categorical data
- Evaluating and comparing the different imputations

Congratulations!!

DEALING WITH MISSING DATA IN PYTHON