MIS710 Machine Learning in Business
Topic 6: K Nearest Neighbors

Associate Professor Lemai Nguyen

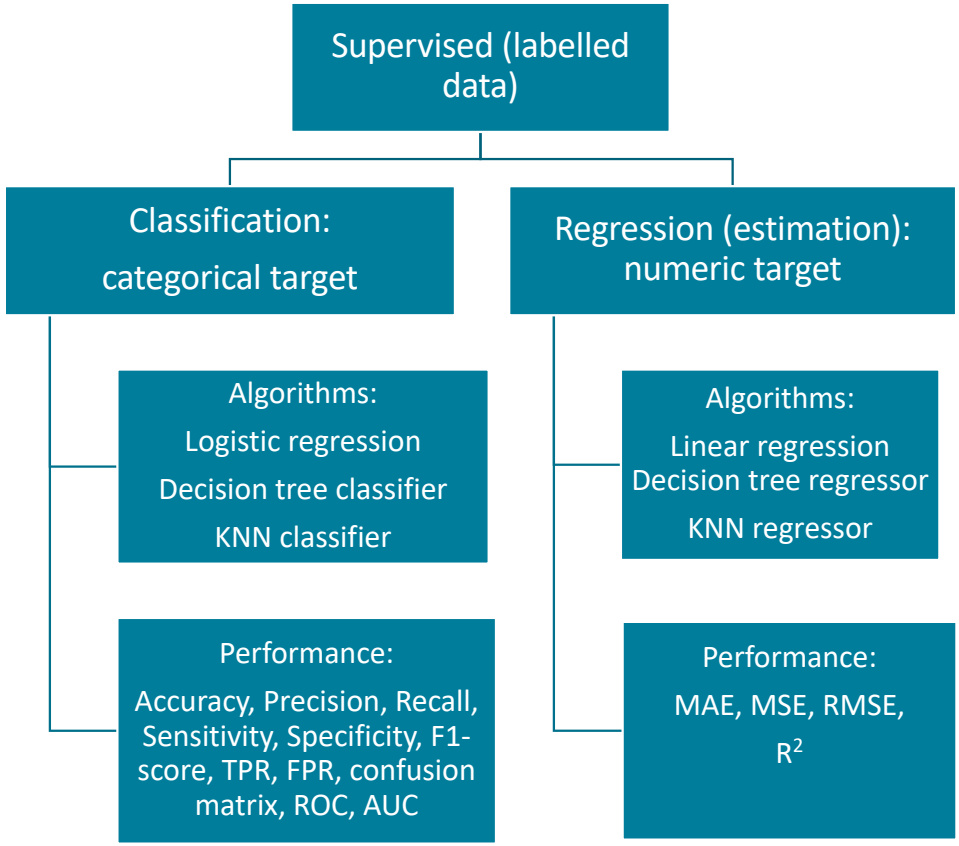
Learning Analytics for Primary Schools

A2

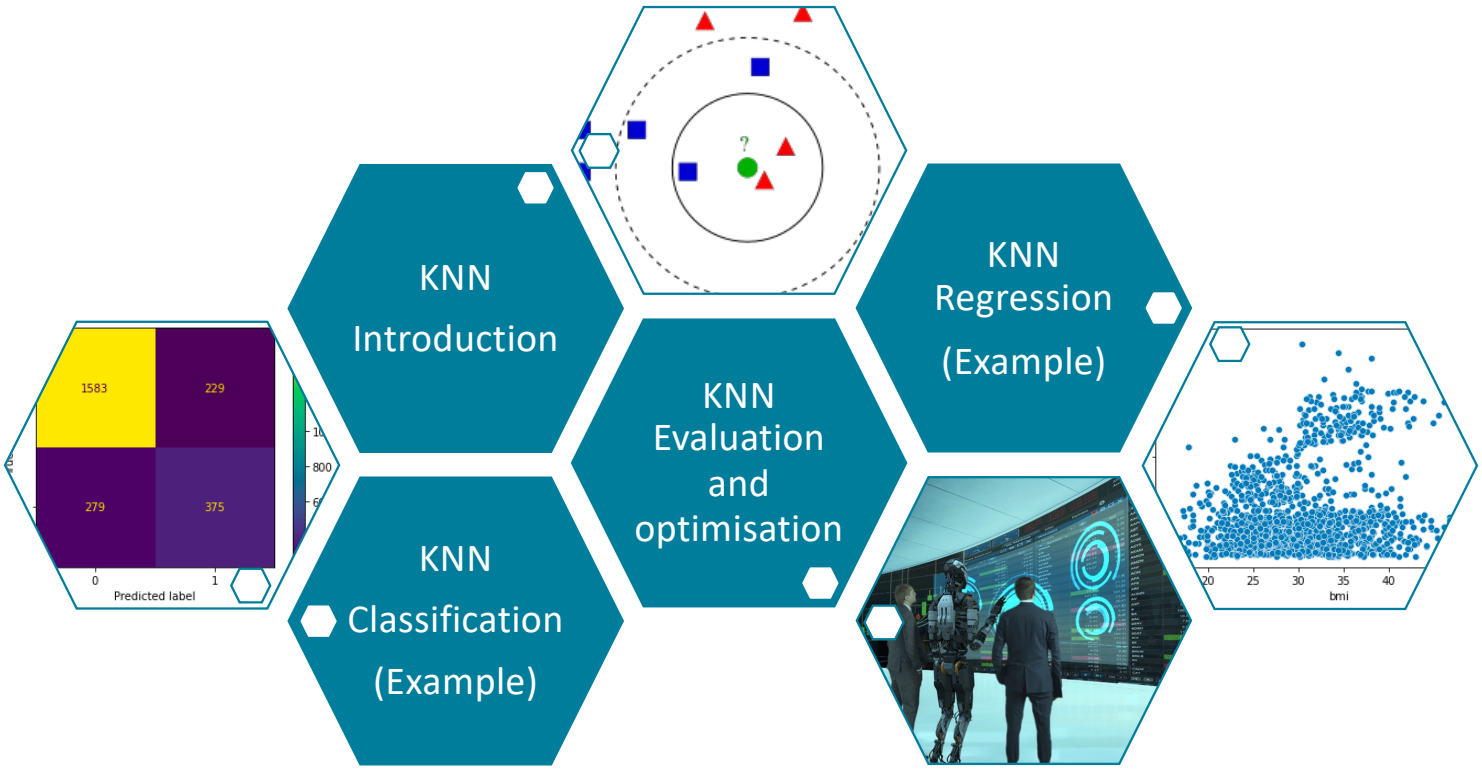
Case Study

A2 Tasks

A2 Deliveries


Supervised (labelled data)
Classification: categorical target
Regression (estimation): numeric target
Algorithms:
Logistic regression
Decision tree classifier
KNN classifier
Performance:
Accuracy, Precision, Recall, Sensitivity, Specificity, F1-score, TPR, FPR, confusion matrix, ROC, AUC
Algorithms:
Linear regression
Decision tree regressor
KNN regressor
Performance:
MAE, MSE, RMSE,
R^2

Supervised Machine Learning with KNN

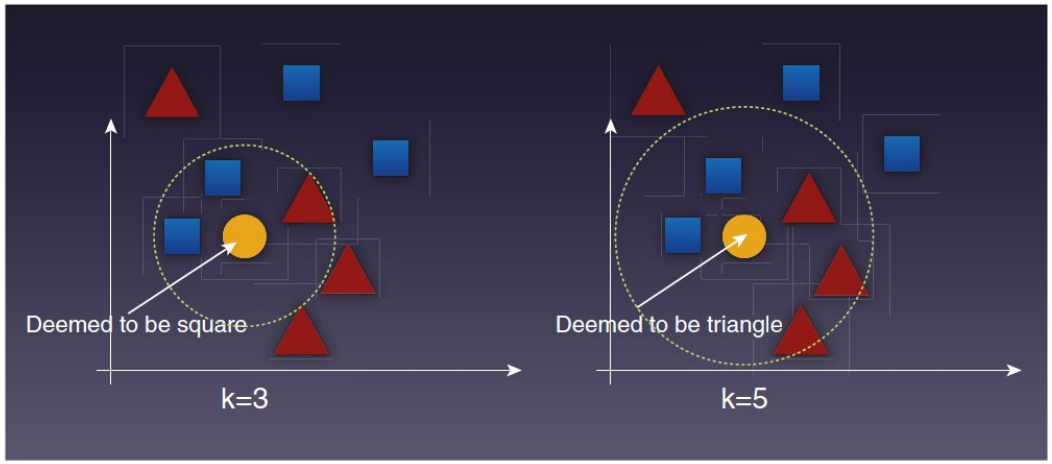


KNN

Introduction

Intuition: Birds of a feather flock together (classification)

- Given a (historical) dataset with labels and a new datapoint subject to prediction (without label)
- Compute the distance between the new datapoint and and each of the datapoints in the given dataset
- Find the nearest K neighbours of the datapoint of interest
- Takes the majority of the classes of these K-neighbours to be the prediction



Wei-Meng Lee, 2019

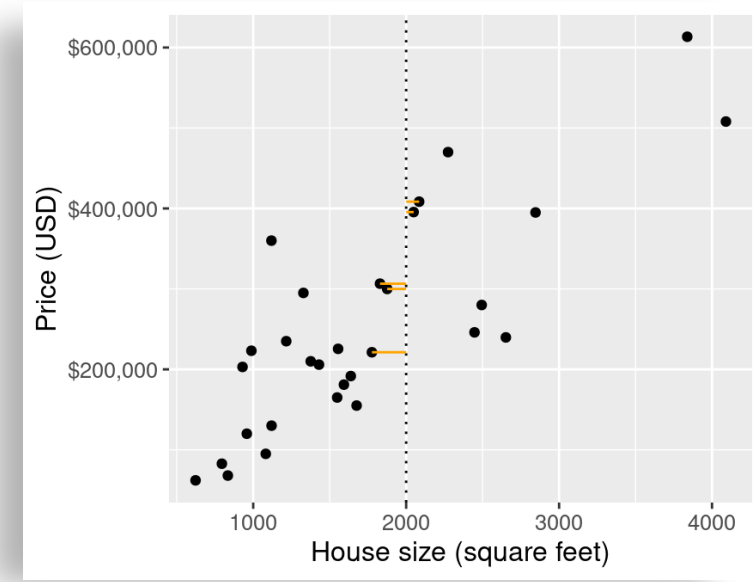
Prediction is sensitive to K

For example, k=3, predict whether the orange data point should be of the class circle or square

Questions: What if k=1?
What if k=N?
What is 'best' k?

Intuitions: Birds of a feather flock together (regression)

- Given a (historical) dataset with labels and a new datapoint subject to prediction (without label)
- Compute the distance between the new datapoint and and each of the datapoints in the given dataset
- Find the nearest K neighbours of the datapoint of interest
- Takes the **mean of these K-neighbours to be the estimation**



Timbers et al, chapter 7, 2022

For example, k=5, estimate the price of a new house of 2000 square feet

Euclidean distance between datapoints

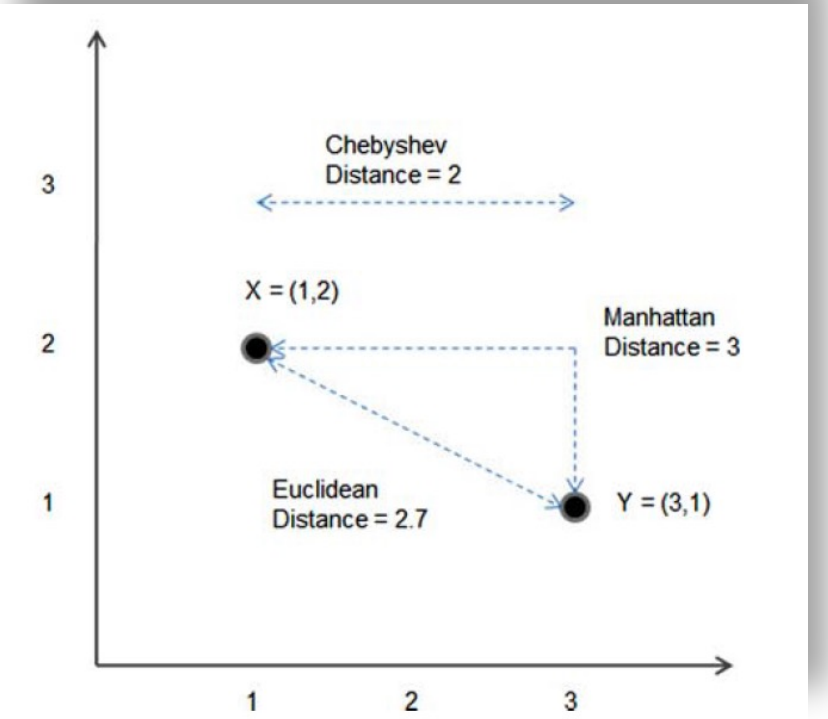
Euclidean distance measure the straight line between X and Y

In a two-dimensional space (2 predictors):

$$\text{Distance } d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

In a n-dimensional space (n predictors)

$$\text{Distance } d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$



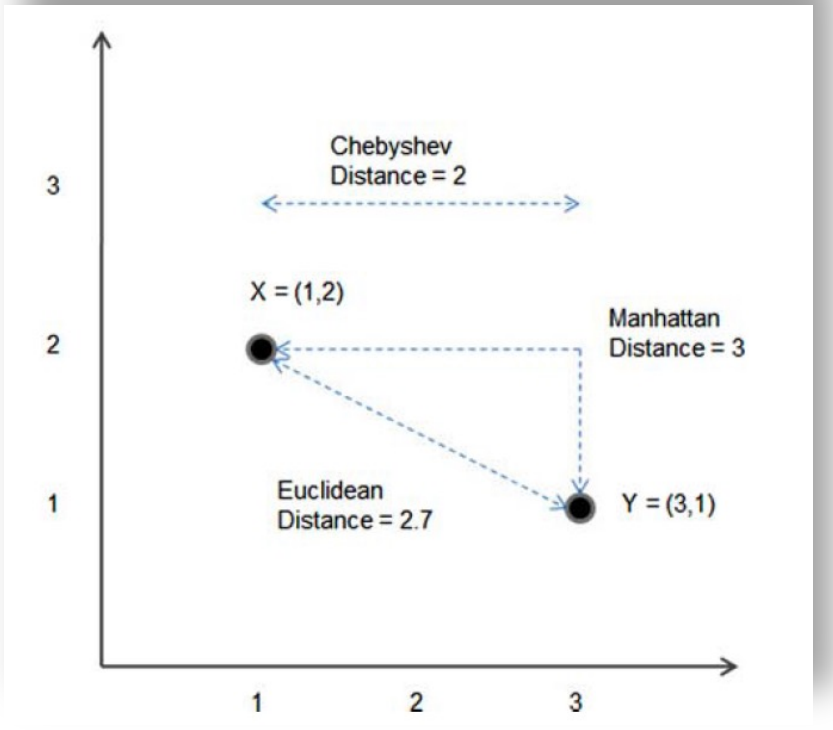
Kotu and Deshpande, 2019, chapter 4

Manhattan distance between datapoints

Manhattan distance between X and Y measures the absolute value between two points, also known as also referred to as taxicab distance in a city grid.

For example, in a n-distance space (n predictors):

$$\text{Distance d} = \sum_{i=1}^{n} |x_i - y_i|$$



Kotu and Deshpande, 2019, chapter 4

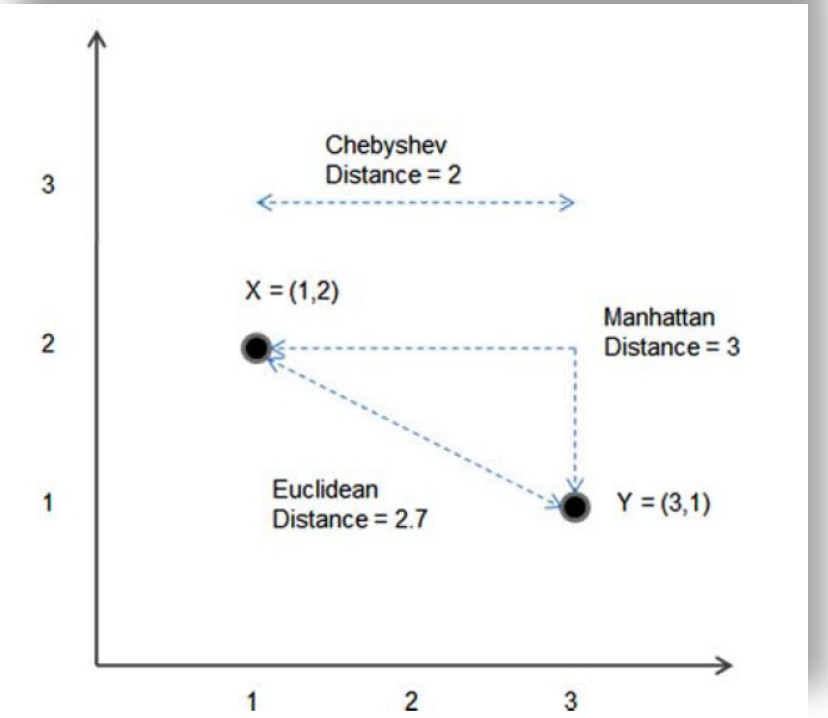
Minkowski distance between datapoints

Minkowski distance between X and Y is is the generalised form of Euclidean and Manhattan distance metrics.

For example, in a n-dimensional space (n predictors):

$$d = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

p=1 then d is Manhattan distance
P=2 then d is Euclidean distance



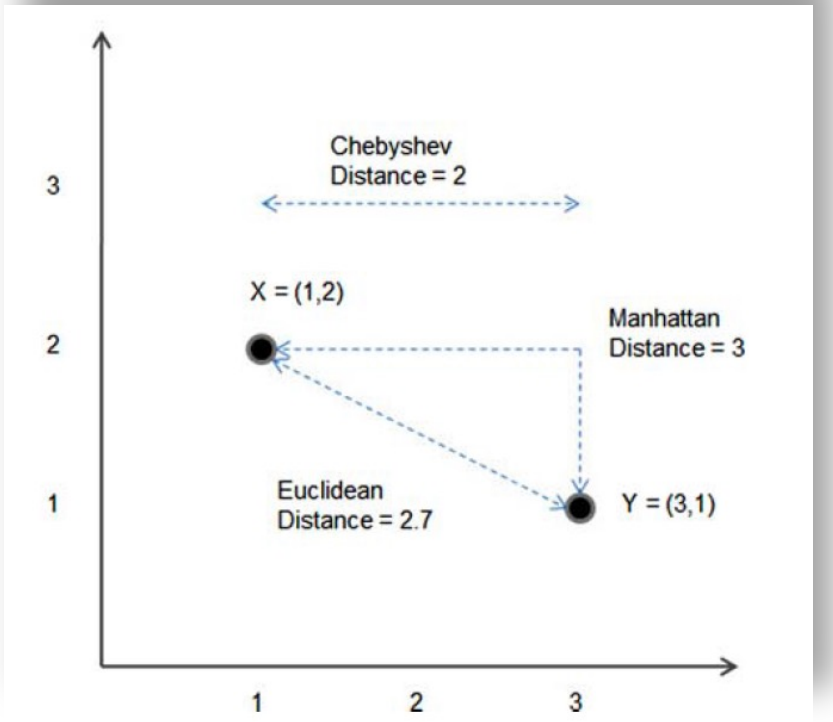
Kotu and Deshpande, 2019, chapter 4

Chebyshev distance between datapoints

Chebyshev distance between X and Y is is the greatest of the differences along any coordinate dimension.

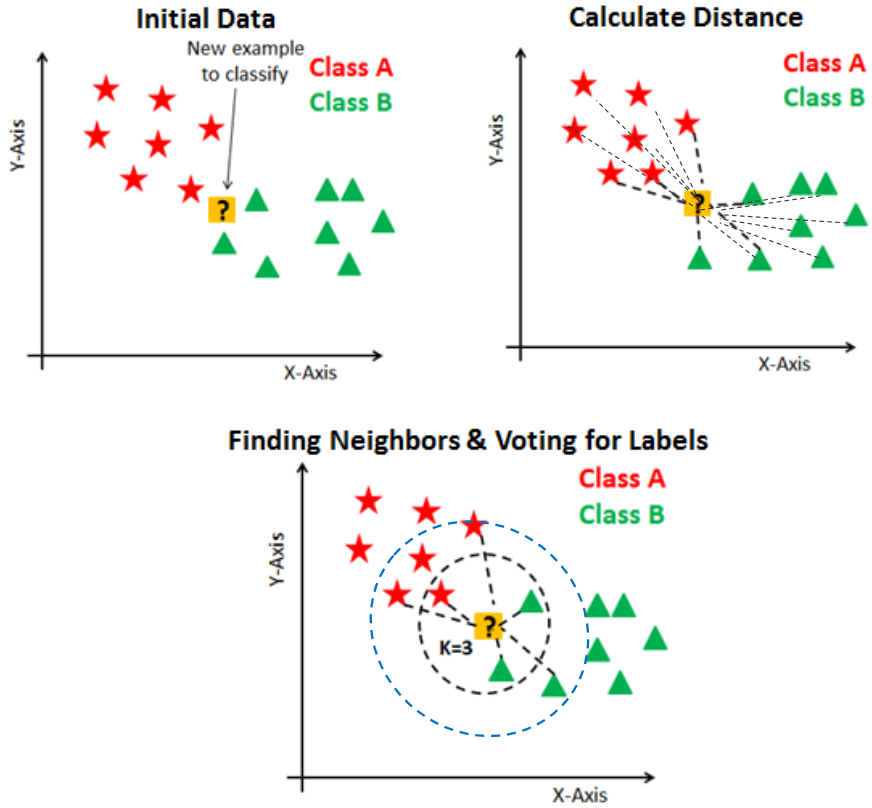
For example, in a n-dimensional space (n predictors):

$$\text{Distance d} = \max_{i=1\ to\ n} |xi - yi|$$



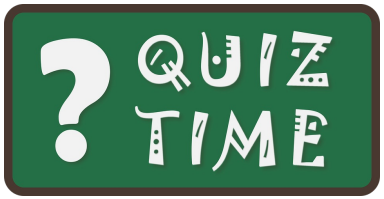
Kotu and Deshpande, 2019, chapter 4

Implementing KNN algorithm



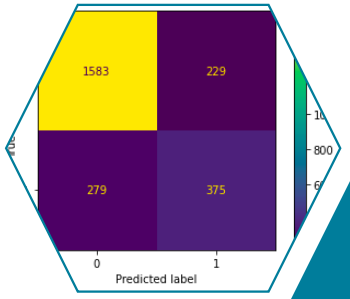
Adapted https://towardsdatascience.com/knn-visualization-in-just-13-lines-of-code-32820d72c6b6

K-nearest neighbors (KNN) - a lazy leaner



- A **lazy** learner memorises/stores the training set, delays computation until making predictions
  - Less time training, more time predicting
  - KNN: instance-based

- In contrast, an eager learner (e.g., linear regression, decision tree) creates a relationship model during training
  - More time training, less time predicting



KNN Classification (Example)

Machine Learning

Kotu and Deshpande, 2019, chapter 1

Supervised machine learning

- infer a model or relationship based on labelled training data
- use this model to map new unlabelled data to predict output variables
- classification and estimation

ExampleSet (7043 examples, 0 special attributes, 11 regular attributes) Filter (7,043 / 7,043 examples): all

Row No.	customerID	gender	SeniorCitiz...	PhoneServi...	MultipleLin...	TechSupport	Streaming...	PaymentMe...	MonthlyCh...	TotalCharg...	Churn
484	5168-MQQCA	Female	0	Yes	Yes	Yes	Yes	Bank transfe...	108.500	8003.800	No
485	5949-XIKAE	Female	0	Yes	No	No	No	Electronic ch...	83.550	680.050	Yes
486	7971-HLVXI	Male	0	Yes	Yes	No	Yes	Credit card ...	84.500	6130.850	No
487	9094-AZPHK	Female	0	Yes	Yes	No	Yes	Electronic ch...	100.150	1415	No
488	3649-JPUGY	Male	0	Yes	Yes	Yes	Yes	Bank transfe...	88.600	6201.950	No
489	4472-LVYGI	Female	0	No	No phone s...	Yes	No	Bank transfe...	52.550	?	No
490	8372-JUXUI	Male	0	Yes	Yes	No	No	Electronic ch...	74.350	74.350	Yes
491	3552-CTCYF	Male	0	Yes	Yes	No	Yes	Bank transfe...	104.800	6597.250	No
492	6778-YSNIH	Female	0	Yes	No	No	No	Electronic ch...	59	114.150	No
493	0388-EOPEX	Female	0	Yes	No	No	No	Electronic ch...	74.400	139.400	Yes
494	5756-OZRIO	Male	1	Yes	Yes	No	Yes	Bank transfe...	64.050	3902.600	No
495	6579-JPICP	Male	0	Yes	No	No internet ...	No internet ...	Mailed check	20.400	20.400	No
496	8205-OTCHB	Male	0	No	No phone s...	No	Yes	Bank transfe...	43.750	903.600	Yes

https://www.kaggle.com/pavanraj159/telecom-customer-churn-prediction

churn probability

loyal probability

Machine Learning in Business Framing

- High competition in the market
- Low retention rate!
- Need to predict who is likely to leave to offer incentives

Business needs (BA)

- Customer data from contracts
- Behaviour data from usage and payment data
- Churn data

Data

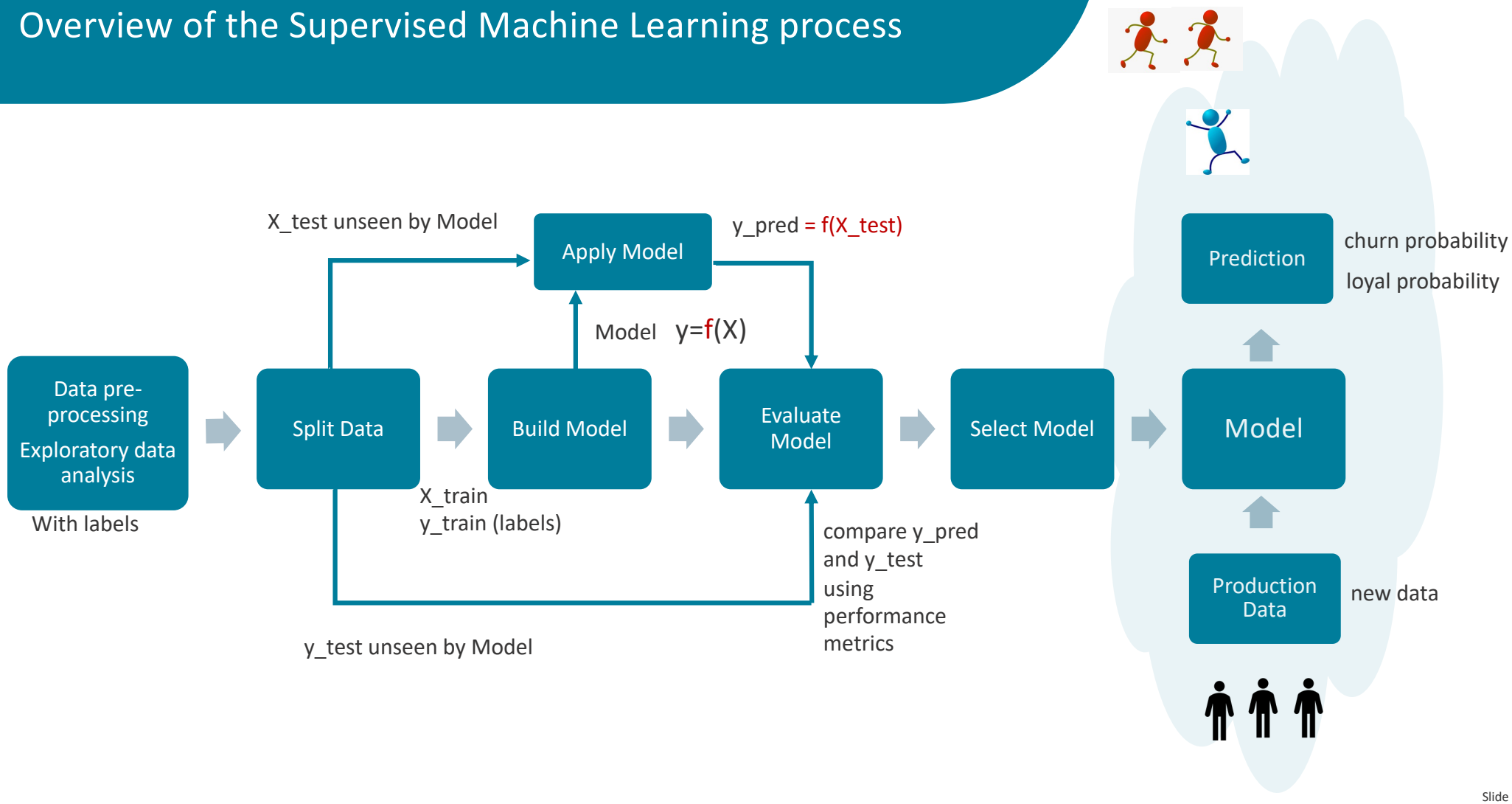
- Predictive analytics to classify customers into Churn or Loyal
- Supervised, classification

Analytics (ML)

- Customer support process (Call centre, Email enquiries)
- Post-sales: Promotion and engagement processes

Deployment (Serving)

Overview of the Supervised Machine Learning process



Data preparation (1)

```
#loading data
records = pd.read_csv('https://raw.githubusercontent.com/VanLan0/MIS710/main/Customers.csv')
```

```
#data types
records.info()
```

```
#Inspect missing data
print(records.isnull().sum())
```

```
#visualise the column with missing data
sns.histplot(data=records, x='TotalCharges',
bins=20, kde=True)
```

```
#draw the boxplot for the column with missing data
sns.boxplot(x=records['TotalCharges'], showmeans =
True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

Data preparation (2)

```
#loading data
records = pd.read_csv('https://raw.githubusercontent.com/VanLan0/MIS710/main/Customers.csv')
```

```
#data types
records.info()
```

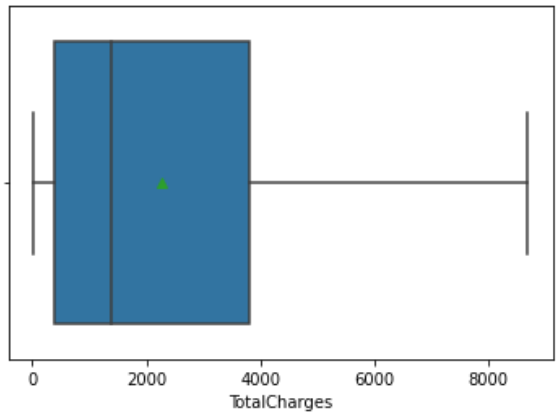
```
#Inspect missing data
print(records.isnull().sum())
```

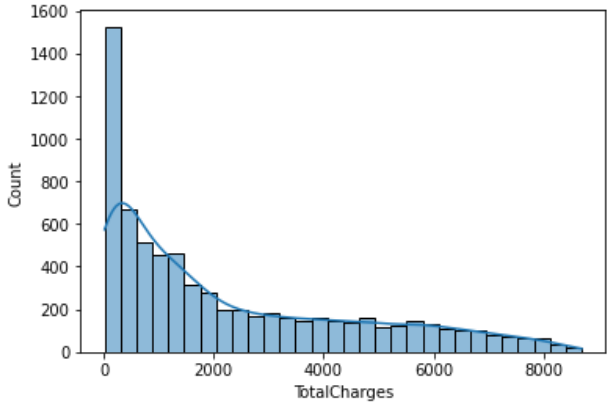
There are 11 missing values in the column TotalCharges

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

Data preparation (3)

```
#visualise the column with missing data
sns.histplot(data=records, x='TotalCharges', bins=30, kde=True)

#draw the boxplot for the column with missing data
sns.boxplot(x=records['TotalCharges'], showmeans = True)
```






```
#As the distribution is skewed, replace the missing values with median
records['TotalCharges'].fillna(records['TotalCharges'].median(),inplace=True)
```

Data preparation (4)

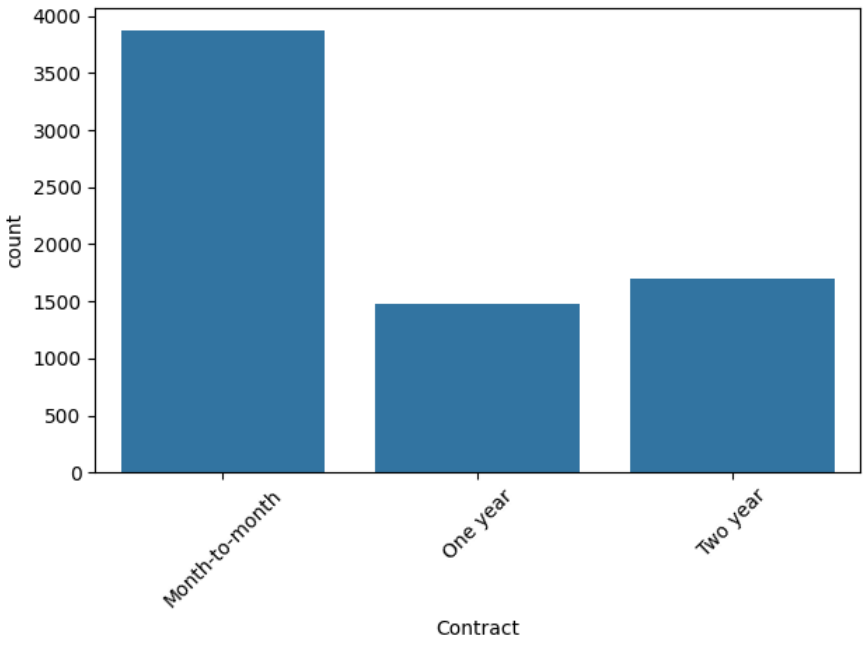
Nominal data

```
#drop irrelevant columns: customer ID
records = records.drop('customerID', axis=1)


# List the categorical columns, on the assumption that they Nominal, are not
Ordinal variables.
cat_columns=['gender', 'Partner', 'Dependents', 'PhoneService',
'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup',
'DeviceProtection',
'TechSupport', 'StreamingTV', 'StreamingMovies', 'PaperlessBilling',
'PaymentMethod', 'Churn']


#Convert categorical variables to numerical using get dummies
records=pd.get_dummies(records, columns=cat_columns, drop_first=True)
```

records

	SeniorCitizen	tenure	Contract	MonthlyCharges	TotalCharges	gender_Male	P			
0	0	1	Month-to-month	29.85	29.85	False				
1	0	34	One year	56.95	1889.50	True				
2	0	2	Month-to-month	53.85	108.15	True				
3	0	45	One year	42.30	1840.75	True				
4	0	2	Month-to-month	70.70	151.65	False				
...	...	...	...	...	...	...				
7038	0	24	One year	84.80	1990.50	True	True	True	True	
7039	0	72	One year	103.20	7362.90	False	True	True	True	
7040	0	11	Month-to-month	29.60	346.45	False	True	True	False	
7041	1	4	Month-to-month	74.40	306.60	True	True	False	True	
7042	0	66	Two year	105.65	6844.50	True	False	False	True	

7043 rows × 30 columns



Data preparation (4)

Ordinal data

```
#Convert categorical variables to numerical using get dummies
records2=pd.get_dummies(records, columns=['Contract'])
```

records2

PaperlessBilling_Yes	PaymentMethod_Credit card (automatic)	PaymentMethod_Electronic check	PaymentMethod_Mailed check	Churn_Yes	Contract_Month-to-month	Contract_One year	Contract_Two year
True	False	True	False	False	True	False	False
False	False	False	True	False	False	True	False
True	False	False	True	True	True	False	False
False	False	False	False	False	False	True	False
True	False	True	False	True	True	False	False
...	...	...	...	...	...	...	...
True	False	False	True	False	False	True	False
True	True	False	False	False	False	True	False
True	False	True	False	False	True	False	False
True	False	False	True	True	True	False	False
True	False	False	False	False	False	False	True

Data preparation (4)

Ordinal data

```
#Convert categorical variables to numerical using get dummies
records3=pd.get_dummies(records, columns=['Contract'], drop_first=True)
```

StreamingMovies_Yes	PaperlessBilling_Yes	PaymentMethod_Credit card (automatic)	PaymentMethod_Electronic check	PaymentMethod_Mailed check	Churn_Yes	Contract_One year	Contract_Two year
False	True	False	True	False	False	False	False
False	False	False	False	True	False	True	False
False	True	False	False	True	True	False	False
False	False	False	False	False	False	True	False
False	True	False	True	False	True	False	False
...	...	...	...	...	...	...	...
True	True	False	False	True	False	True	False
True	True	True	False	False	False	True	False
False	True	False	True	False	False	False	False
False	True	False	False	True	True	False	False
True	True	False	False	False	False	False	True

Data preparation (5)

Ordinal data

```
# Mapping ordinal variable Contract
contract_mapping = {
'Month-to-month': 1,
'One year': 2,
'Two year': 3
}

# Apply the mapping to create a new column 'Contract_N'
records['Contract'] = records['Contract'].map(contract_mapping)
```

records

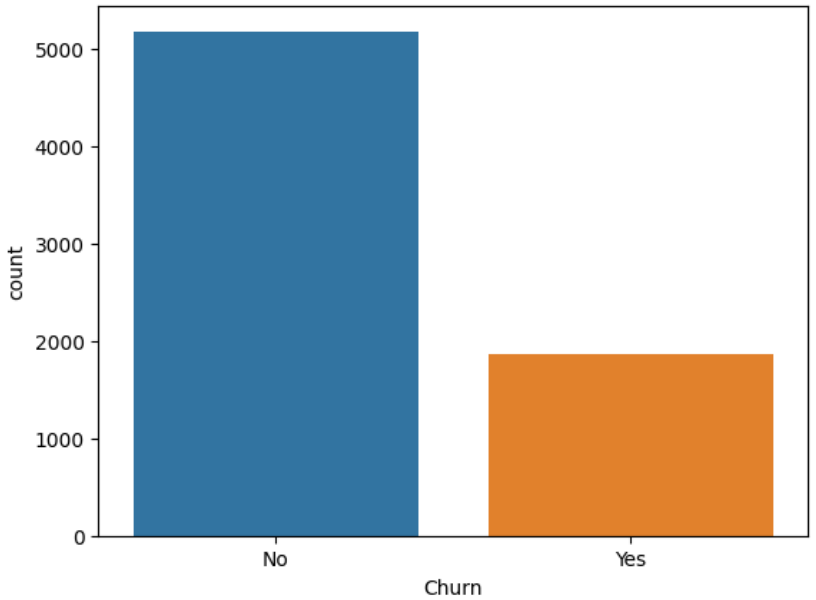
	SeniorCitizen	tenure	Contract	MonthlyCharges	Tot
0	0	1	1	29.85	
1	0	34	2	56.95	
2	0	2	1	53.85	
3	0	45	2	42.30	
4	0	2	1	70.70	
...	...	...	...	...	
7038	0	24	2	84.80	
7039	0	72	2	103.20	
7040	0	11	1	29.60	
7041	1	4	1	74.40	
7042	0	66	3	105.65	

7043 rows × 30 columns

Exploratory data analysis

- Univariate analysis
- Bivariate analysis
- Multivariate analysis

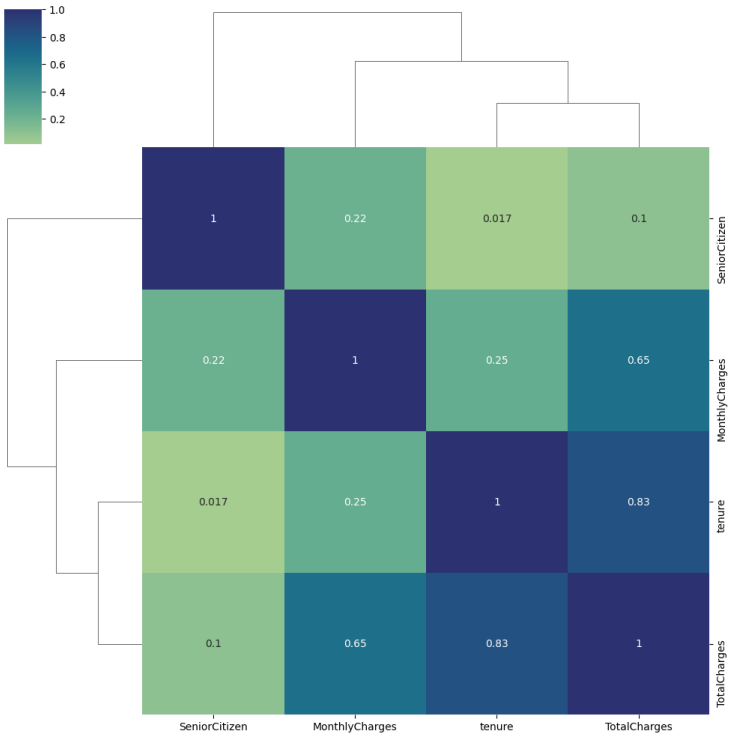
```
#Inspect target variable
sns.countplot(x=records['Churn'])
```



Exploratory data analysis

- Univariate analysis
- Bivariate analysis
- Multivariate analysis

```
sns.clustermap(data=records.corr(),
annot=True, cmap='crest')
```



Select features and target, and split data

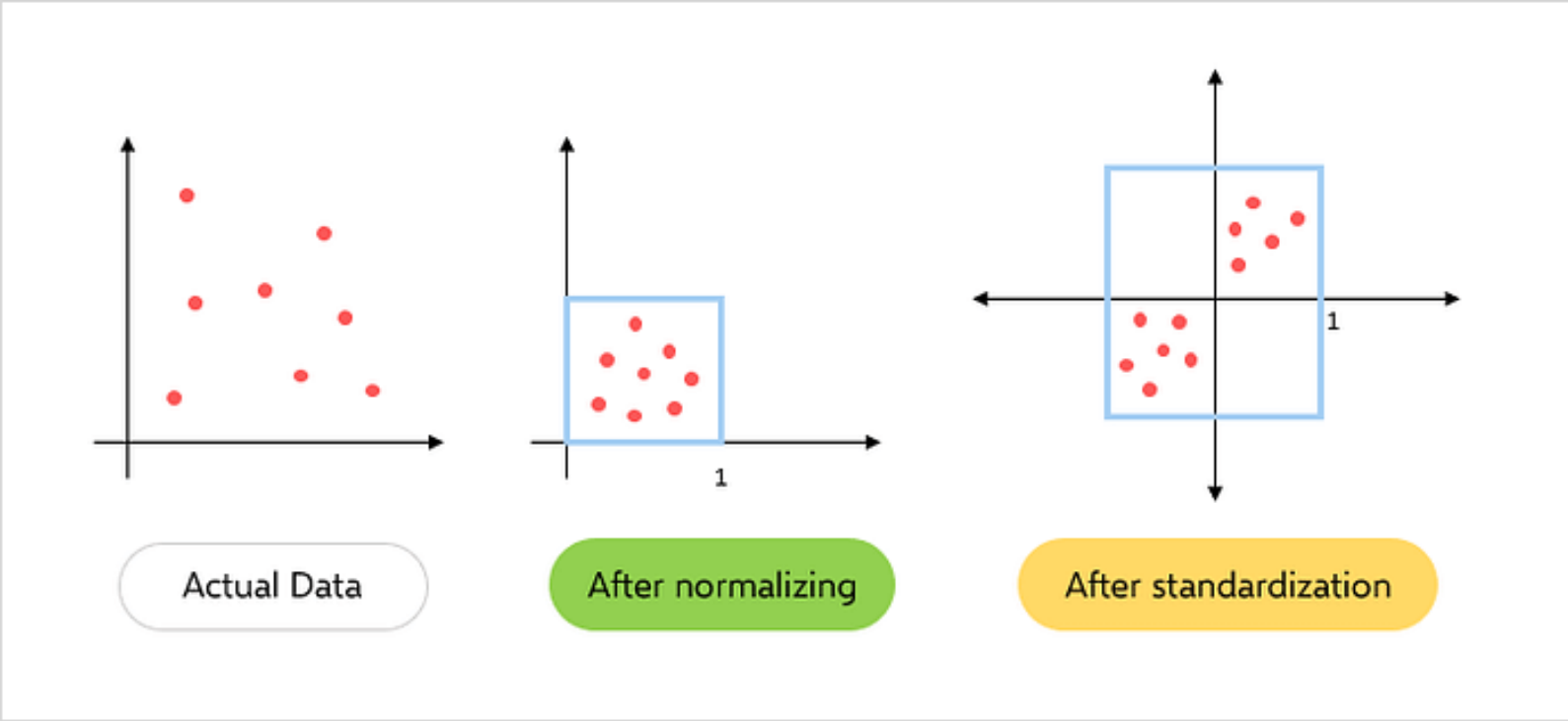
```
#Define predictors and label
X=records.drop('Churn', axis=1)
y=records['Churn']

#import train_test_split
from sklearn.model_selection import train_test_split

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, stratify = y )
```

- Domain knowledge
- Data driven

Scale data

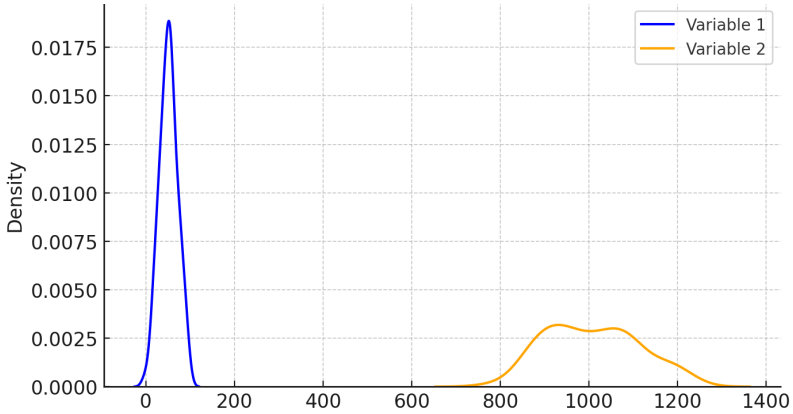
- Algorithms such as decision tree regressors and random forests are generally not very sensitive to scale differences.
- Algorithms such as linear regression and neural networks are sensitive to scale differences.
- Scale **after data split** so that the test data distribution does not affect the scaling of the training process.
- It's common practice not to scale the target variable (y_train and y_test), However, specific models, such as deep learning models, might benefit from scaled targets. In general, scaling the target variable should be done carefully.
- If scaling the target variable (y_train and y_test), you need to perform an inverse transformation on the predicted values to get them back to their original scale.

Scale data



https://becominghuman.ai/what-does-feature-scaling-mean-when-to-normalize-data-and-when-to-standardize-data-c3de654405ed

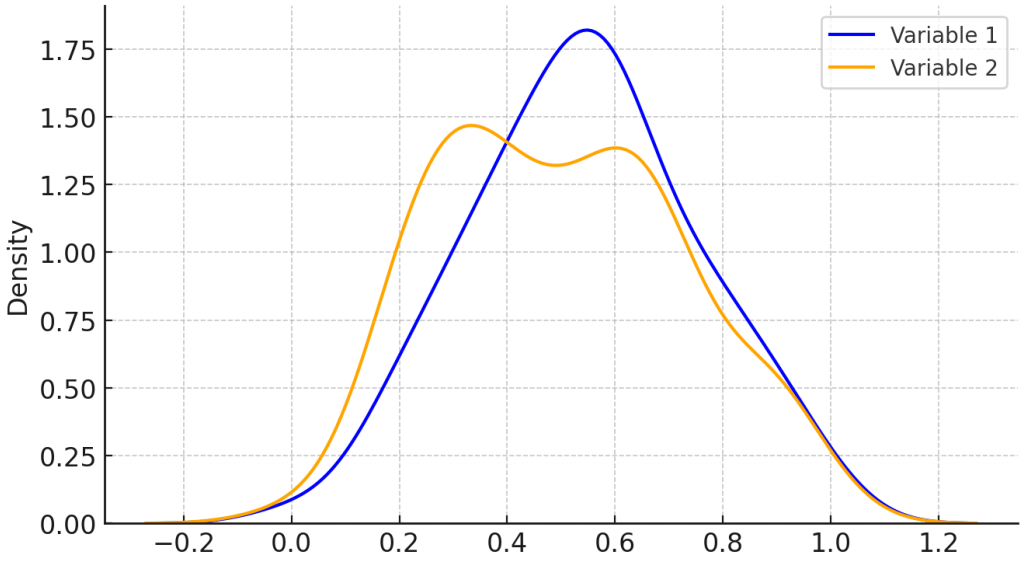
Scale data



Min-Max Scaling rescales a feature to a specific range e.g. [0,1] by subtracting the minimum value of the feature and then dividing by the range of the feature.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

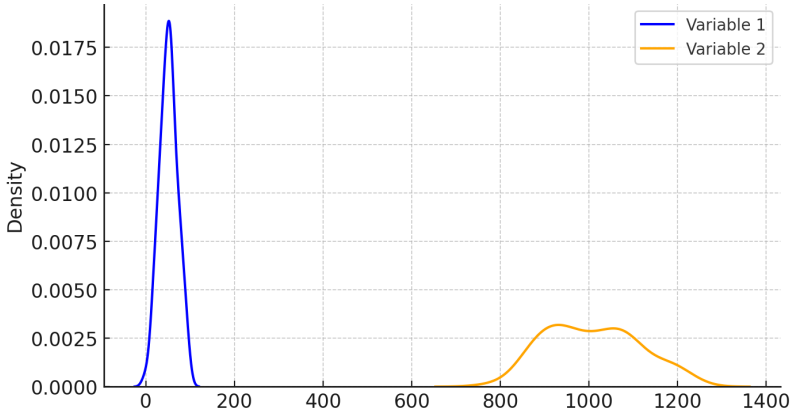
does not assume normal distribution



Advantage: Often used when a **bounded interval** is needed, e.g. in image processing, input values are typically required to be within a specific range

Disadvantage: **Sensitive to outliers**, extreme values can skew the range and compress the scaled values of majority of the data points.

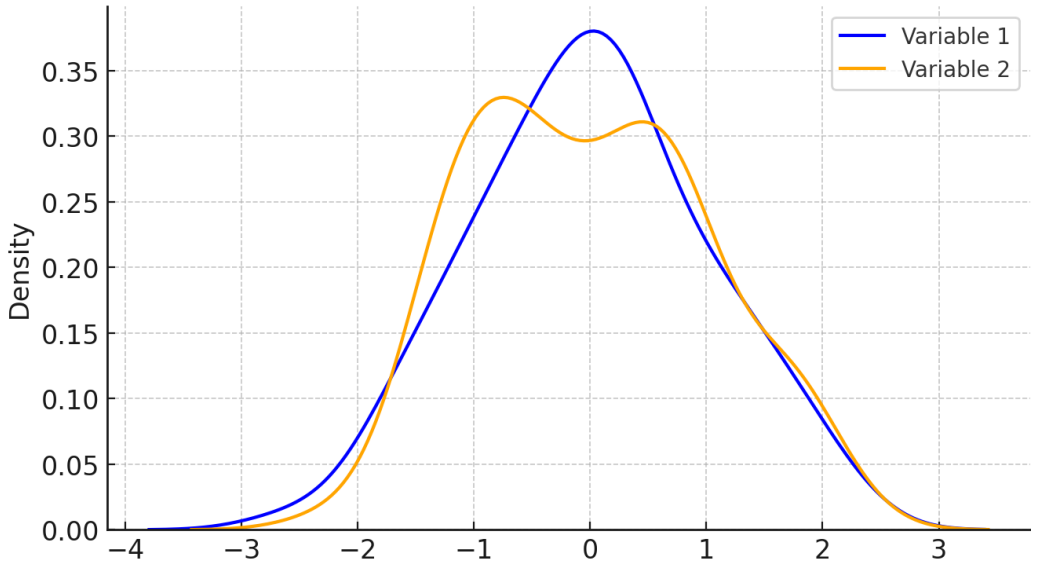
Normalisation (Standard Scaling)



Known as z-score normalisation, transforms the data to have a mean of 0 and a standard deviation of 1.

$$x_{scaled} = \frac{x - mean}{std}$$

assumes normal distribution (e.g., linear regression, logistic regression, SVM).



Advantage: **Less sensitive to outliers** compared to min-max scaling.

Disadvantage: **Does not bound** data within a fixed range.

Scale data using MinMaxScaler

```
#import scaler
from sklearn.preprocessing import MinMaxScaler

# Scaling the features using MinMaxScaler
scaler = MinMaxScaler()

# Fit the scaler on the training data
X_train_scaled = scaler.fit_transform(X_train)

# Transform the test data using the same scaler
X_test_scaled = scaler.transform(X_test)
```

Scale data using StandardScaler

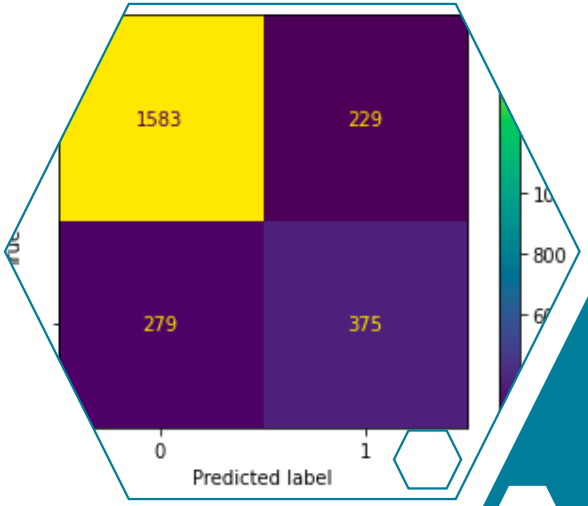
```
#import scaler
from sklearn.preprocessing import StandardScaler

# Normalize the features using StandardScaler
scaler = StandardScaler()

# Fit the scaler on the training data
X_train_scaled = scaler.fit_transform(X_train)

# Transform the test data using the same scaler
X_test_scaled = scaler.transform(X_test)
```

Model building

p parameter for the Minkowski metric.
p = 1 Manhattan distance
for p = 2 Euclidean distance

```
# Import KNN classifier
from sklearn.neighbors import KNeighborsClassifier


# Initialise and Train a KNN model
k = 15 # Number of nearest neighbours
knn = KNeighborsClassifier(n_neighbors=k)

knn.fit(X_train_scaled, y_train)

# Make predictions on the testing set
y_pred = knn.predict(X_test_scaled)


inspection=pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
inspection.sample(10)
```

	Actual	Predicted
1942	False	False
127	False	False
2407	True	True
3314	False	False
3898	False	False
6006	False	False
3446	False	False
3332	True	False
2043	False	True
943	False	False



Evaluation

Evaluation

```
[[1372  180]
 [ 272  289]]
```

	precision	recall	f1-score	support
False	0.83	0.88	0.86	1552
True	0.62	0.52	0.56	561
accuracy			0.79	2113
macro avg	0.73	0.70	0.71	2113
weighted avg	0.78	0.79	0.78	2113

```
# Import metrics and classification report
from sklearn.metrics import classification_report
```

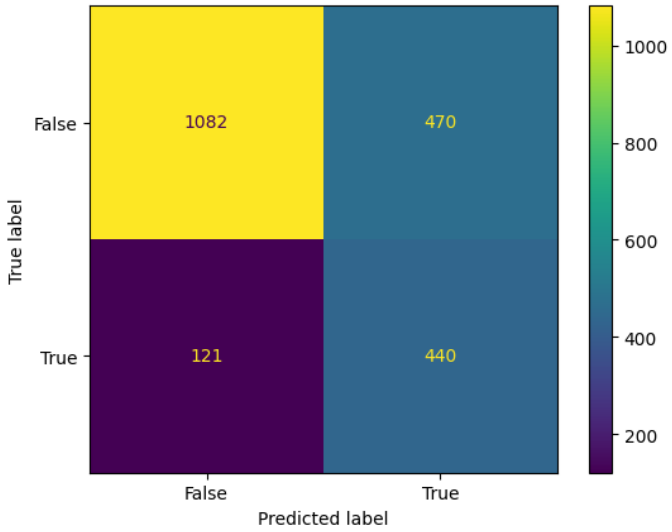
```
# Print the classification report
print(classification_report(y_test, y_pred))
```

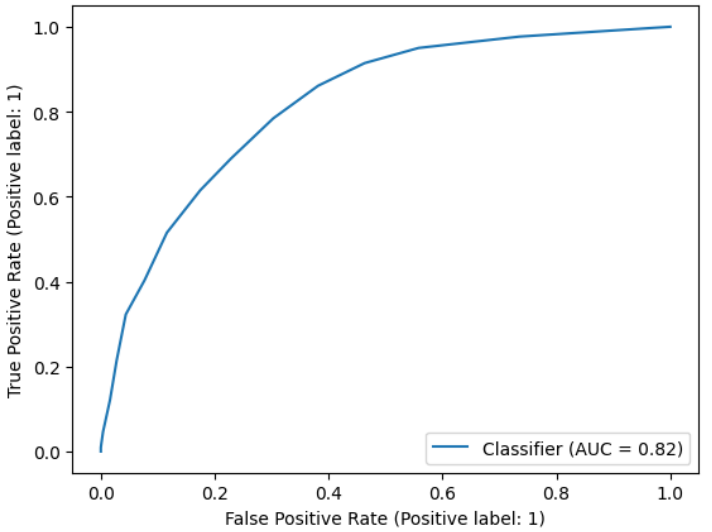
```
#get predicted probabilities for the main class
y_pred_probs_norm = knn.predict_proba(X_test_scaled)
y_pred_probs_norm = y_pred_probs_norm[:, 1]
print(y_pred_probs_norm)
```

```
inspection=pd.DataFrame({'Actual':y_test,
'Predicted':y_pred,
'Probability':y_pred_probs_norm})
```

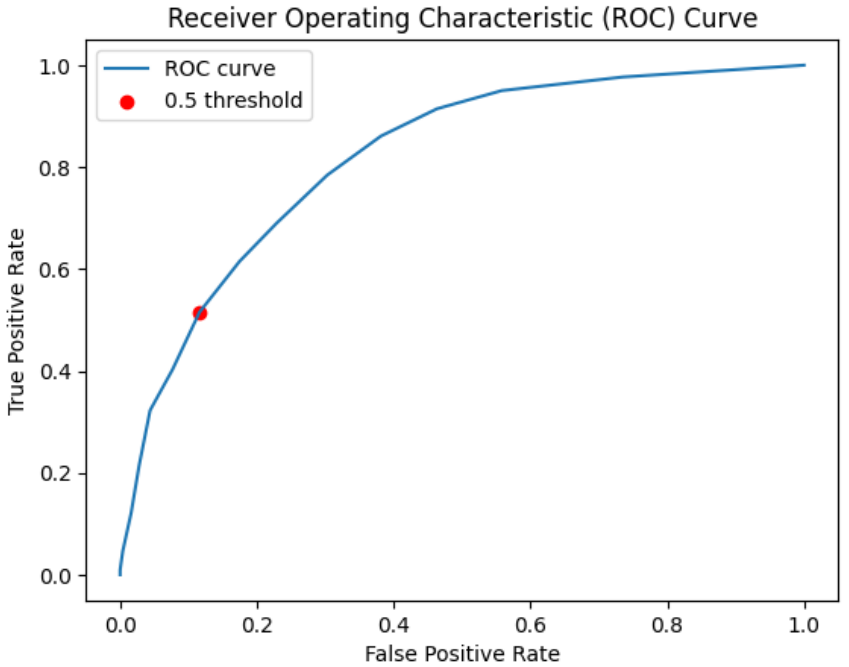
	Actual	Predicted	Probability
4658	False	False	0.333333
2179	False	False	0.066667
4286	False	True	0.600000
2752	True	False	0.133333
3845	True	True	0.533333

```
#import classes to display RocCurve and Confusion Matrix
from sklearn.metrics import RocCurveDisplay
from sklearn.metrics import ConfusionMatrixDisplay

RocCurveDisplay.from_predictions(y_test, y_pred_probs_norm)
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.show()
```





```
# Computer TN FP FN TP
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()

# Compute FPT and TPR for this k
tpr_k = tp / (tp + fn)
fpr_k = fp / (fp + tn)

print('True Positive Rate: ', '%.3f' % tpr_k)
print('False Positive Rate: ', '%.3f' % fpr_k)
```




```
#fpr and tpr for all thresholders and plot the ROC curve
from sklearn.metrics import roc_curve, roc_auc_score

fpr, tpr, thresholds = roc_curve(y_test, y_pred_probs_norm)
print('AUC:', '%.3f' % metrics.auc(fpr, tpr))
```

```
# plot the ROC curve and the best point
plt.plot(fpr, tpr, label='ROC curve')
plt.scatter(x=fpr_k, y=tpr_k, marker='o', color='red', label=0.5 threshold')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()
```

Best threshold

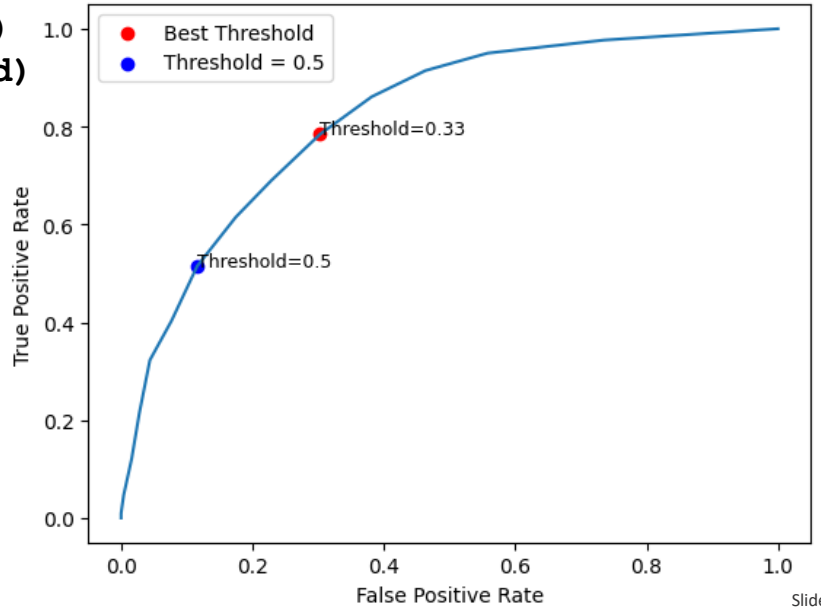
```
from sklearn.metrics import f1_score
# Find the best threshold based on F1 score
f1 = []
for threshold in thresholds:
  y_pred_t = [1 if prob >= threshold else 0 for prob in y_pred_probs_norm]
  f1.append(f1_score(y_test, y_pred_t))
best_threshold = thresholds[f1.index(max(f1))]
```

```
#get predicted probabilities for the best threshold
y_pred_best = (knn.predict_proba(X_test)[:,1] >= best_threshold).astype(bool)
```

Best threshold

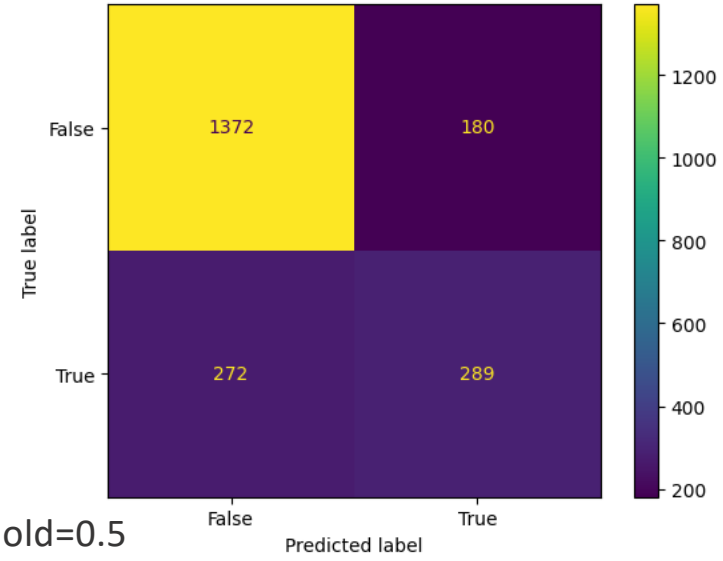
```
# Computer TN FP FN TP for the best threshold
tn_bestThreshold, fp_bestThreshold, fn_bestThreshold, tp_bestThreshold =
confusion_matrix(y_test, y_pred_best).ravel()
print(confusion_matrix(y_test, y_pred_best))
# Compute FPT and TPR for this k
tpr_bestThreshold = tp_bestThreshold / (tp_bestThreshold + fn_bestThreshold)
fpr_bestThreshold = fp_bestThreshold / (fp_bestThreshold + tn_bestThreshold)

print('True Positive Rate: ', '%.3f' % tpr_bestThreshold)
print('False Positive Rate: ', '%.3f' % fpr_bestThreshold)
```

```
[[1082  470]
 [ 121  440]]

True Positive Rate: 0.784
False Positive Rate: 0.303
```

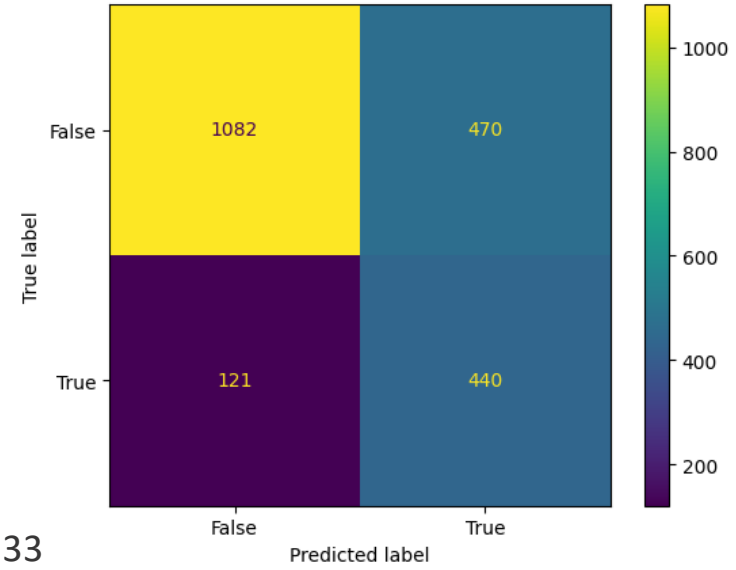


Best threshold

```
#get predicted probabilities for the main class
y_pred_best = (knn.predict_proba(X_test)[:,1] >= 0.27).astype(bool)

ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_best)
```



Threshold=0.5



Threshold=0.33

Model fitting

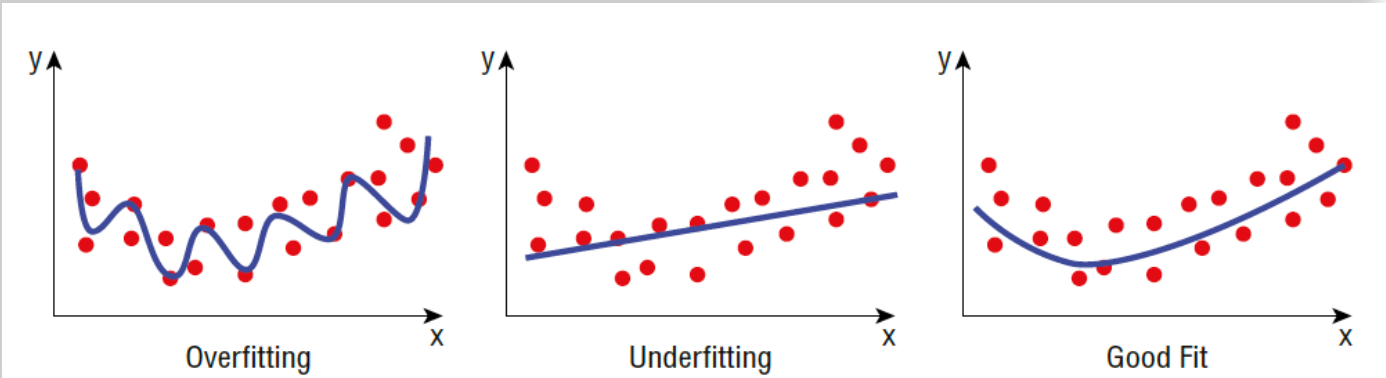
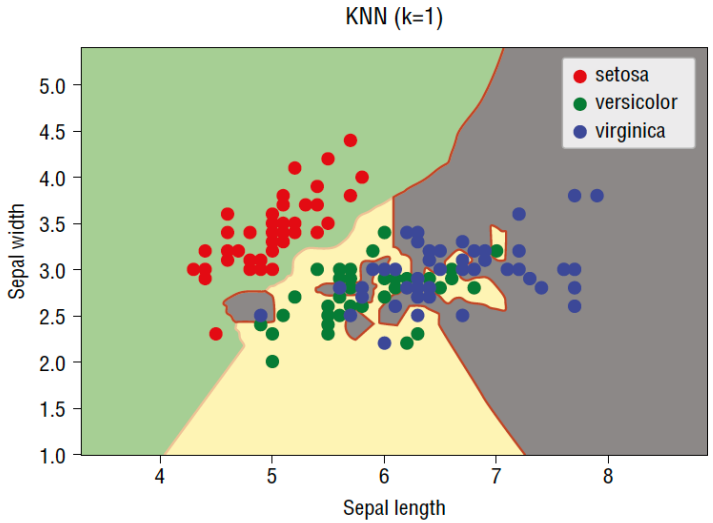


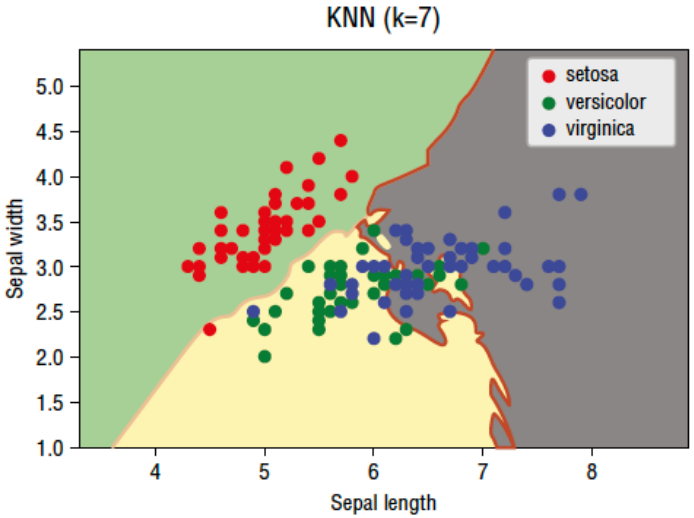
Figure 9.6: Understanding the concept of overfitting, underfitting, and a good fit

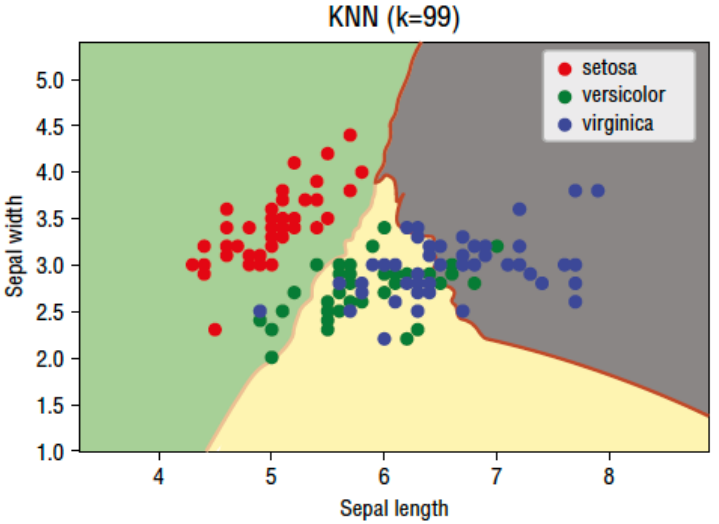
Lee, 2019

Model fitting with KNN



k-1: overfitting – the model fits too well, very sensitive to outliers, mislabelling, will not work well with unseen data.





Setting k too high leads to under fitting, more points will be classified incorrectly.

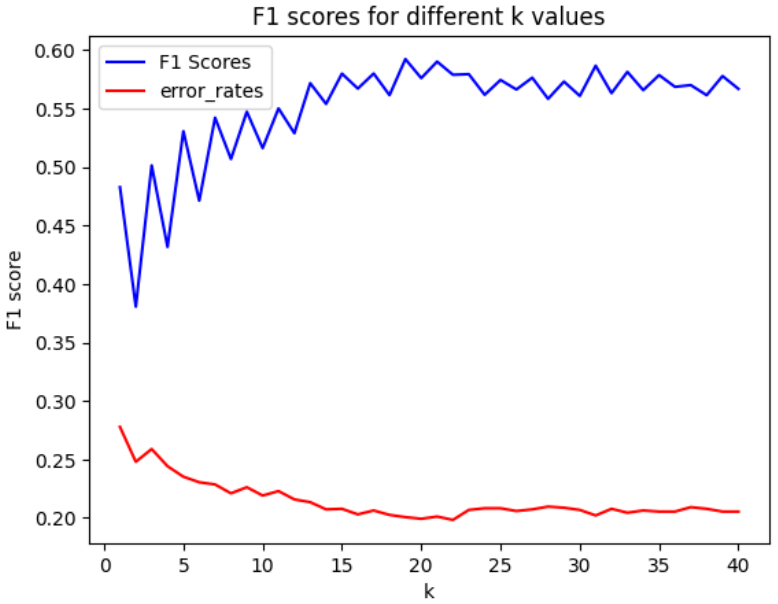
Optimising k

```
# Define a list of k values to test
k_values = list(range(1, 41))

# Train and evaluate KNN classifiers with different k values
best_k=0
best_accuracy=0
best_f1=0
accuracy_scores = []
accuracy = 0
error_rate=1-accuracy
error_rates=[]
for k in k_values:
  knn = KNeighborsClassifier(n_neighbors=k)
  knn.fit(X_train_scaled, y_train)
  y_pred_k=knn.predict(X_test_scaled)
  accuracy = knn.score(X_test_scaled, y_test)
  accuracy_scores.append(accuracy)
  error_rates.append(1-accuracy)
  if accuracy > best_accuracy:
    best_k = k
    best_accuracy = accuracy
    best_f1 = metrics.f1_score(y_test, y_pred_k)
```

Optimising k

```
# Find the best k value with highest accuracy score
print(f"Best k value: {best_k}")
print(f"Best accuracy: {best_accuracy:.3f}")
print(f"F1 score for best accuracy: {best_f1:.3f}")

# Plot k values against accuracy scores
plt.xlabel('k')
plt.ylabel('Error rate')
plt.title('Error rates for different k values')
plt.plot(k_values, error_rates, color='blue')
plt.show()
```

```
Best k value: 18
Best accuracy: 0.795
F1 score for best accuracy: 0.578
```



KNN – Insight for data imputation

- KNNImputer looks at the K nearest neighboring data points that have complete information for a variable with missing and takes an average (or median) of those values to fill in the missing value.
- K is a hyperparameter.

- Note: KNNImputer cannot be directly applied to categorical data.

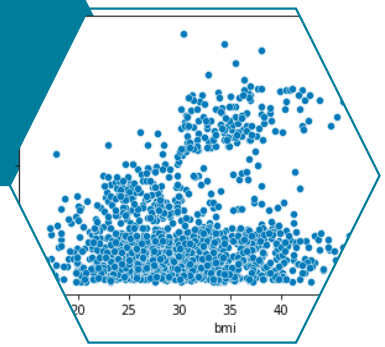
KNN – Insight for data imputation

```
#Previously we replace the missing values with median
records['TotalCharges'].fillna(records['TotalCharges'].median(),inplace=True)
```

Now we can do this:

```
from sklearn.impute import KNNImputer

#Impute missing data using KNN imputation
imputer = KNNImputer(n_neighbors=5)
records[['TotalCharges']] = imputer.fit_transform(records[['TotalCharges']])
```

KNN Regression (Example)



ML in Business Framing: Estimating Health Insurance Premium

- Health insurance company introducing a self-service to support prospective policy holders' decision making and improve their experience; reduce #staff hours to answer quote enquiries
- Need to estimate insurance premium

Business needs (BA)

- Sources:
  - Internal past data: policy dataset
- **Data content**:
  - Policy holder's demographics: age, sex, dependants, region
  - Policy holder's health behaviours: BMI, smoker
  - Label: charger label

Data

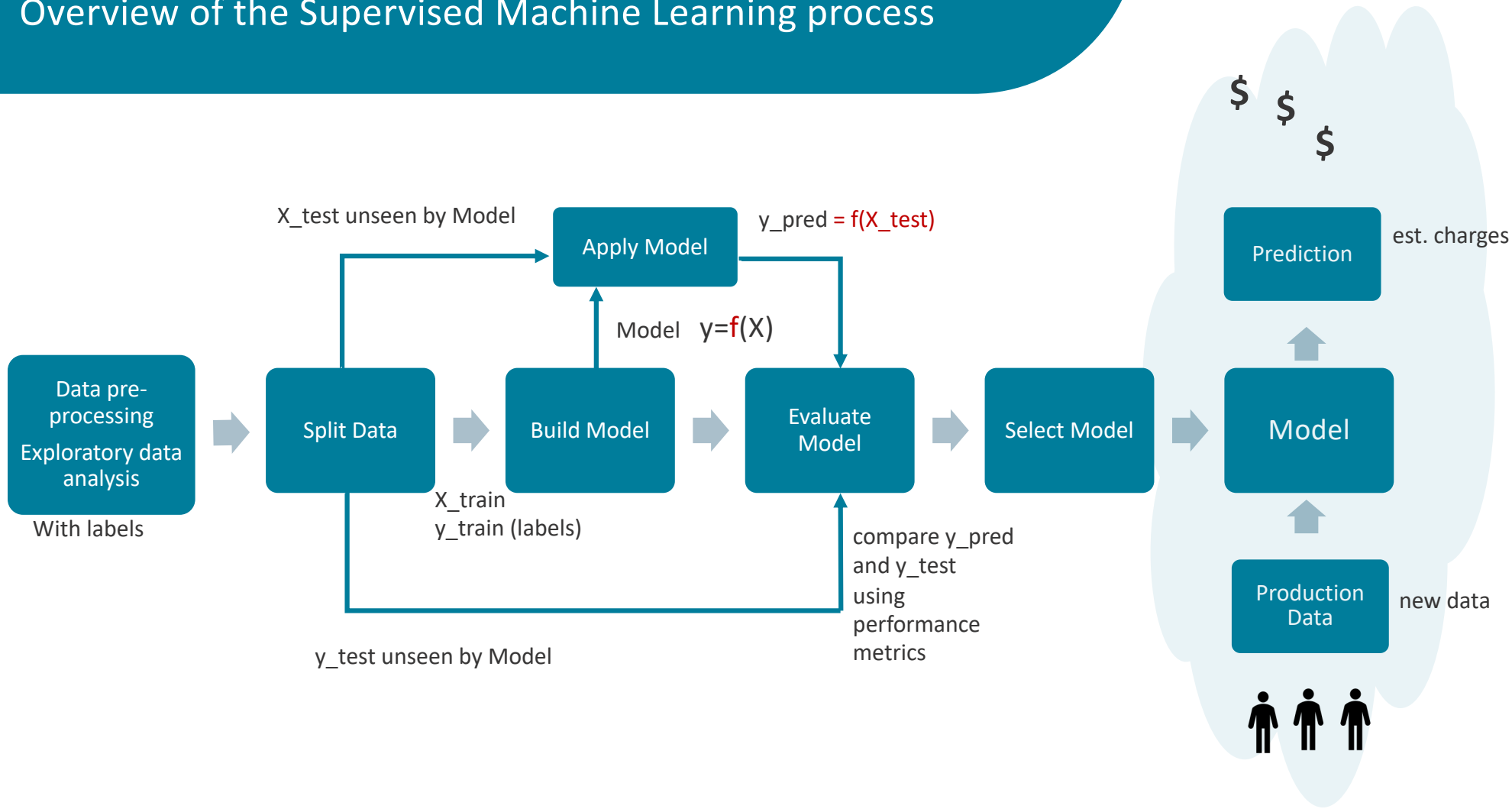
- **Supervised ML, estimation**
- Build and test a **KNN** model that best fit the existing policy holders demographics, health behaviours, and premium values

Analytics (ML)

- Get a Quote self-service tool on website

Deployment (Serving)

Overview of the Supervised Machine Learning process



Loading data

```
#loading data
records = pd.read_csv('https://raw.githubusercontent.com/VanLan0/MIS710/main/insurance.csv')
```

```
records.head()
```

	age	sex	bmi	dependants	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

```
records.info()
```

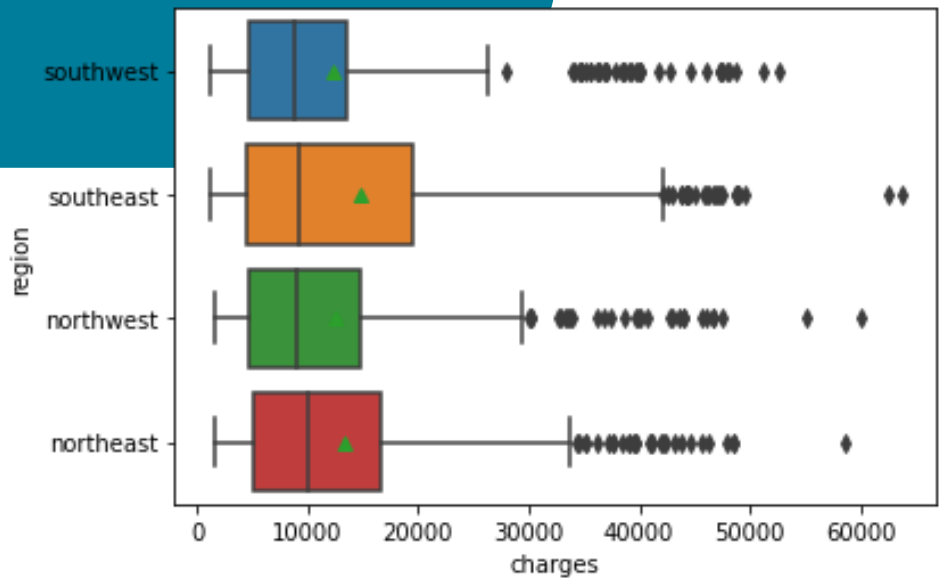
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   age         1338 non-null   int64
 1   sex         1338 non-null   object
 2   bmi         1338 non-null   float64
 3   dependants  1338 non-null   int64
 4   smoker      1338 non-null   object
 5   region      1338 non-null   object
 6   charges     1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

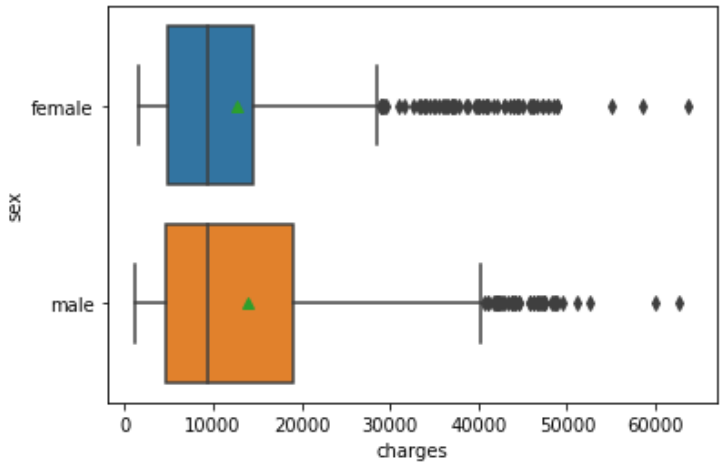
```
#Inspect missing data
print(records.isnull().sum())
```

No missing data

Exploratory data analysis

	age	bmi	dependants	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

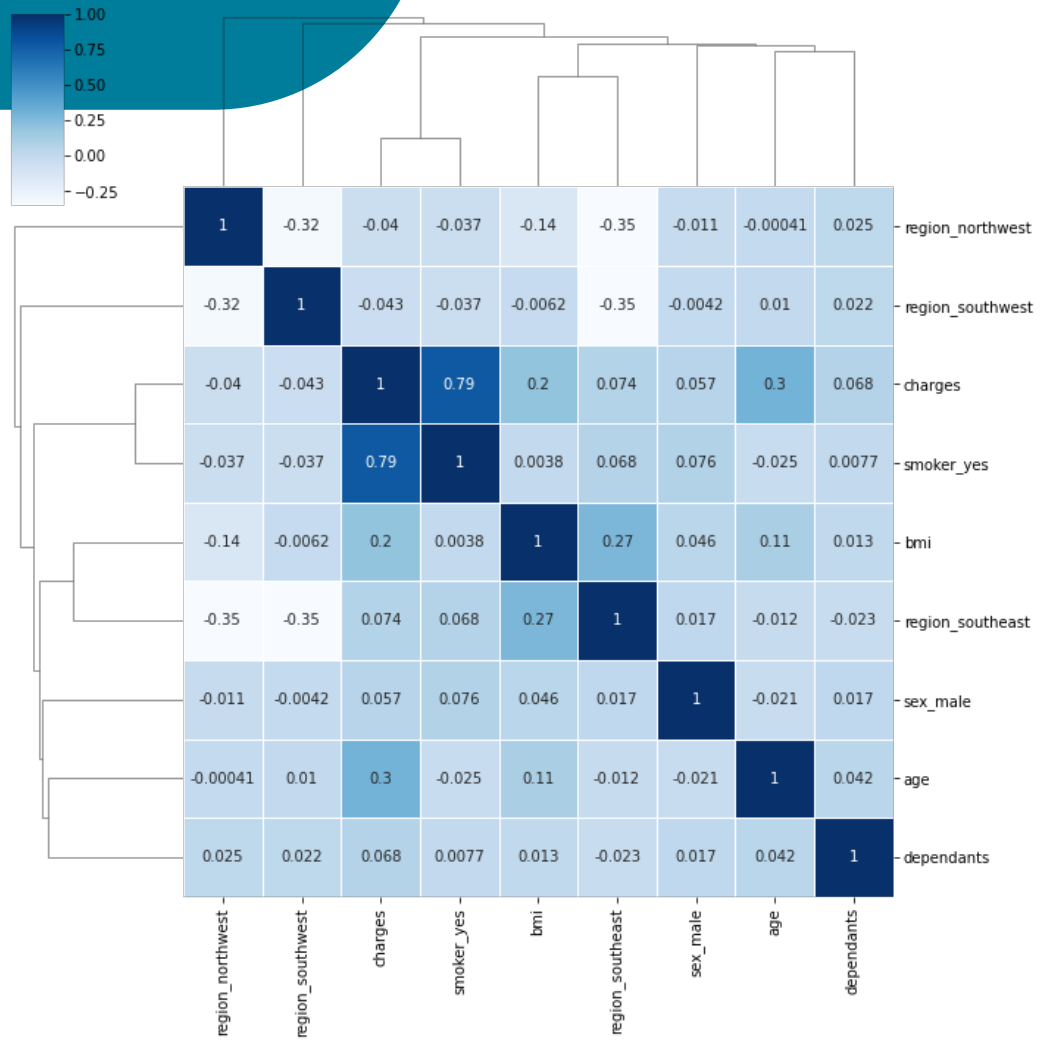




Exploratory data analysis

```
#Convert categorical variables to
numerical using get dummies
records=pd.get_dummies(records,
columns=['sex', 'region'],
drop_first=True)
```

```
#convert categorical data to numerical
def coding_smoking(x):
if x=='yes': return 1
if x=='no': return 0

records['smoker'] =
records['smoker'].apply(coding_smoking)
```



Data preparation and splitting

```
X=records.drop('charges', axis=1)
y=records['charges']

#split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=2024)
```

```
#import scaler
from sklearn.preprocessing import StandardScaler

# Normalize the features using StandardScaler
scaler = StandardScaler()


# Fit the scaler on the training data
X_train_scaled = scaler.fit_transform(X_train)

# Transform the test data using the same scaler
X_test_scaled = scaler.transform(X_test)
```

Fit a model and make predictions

```
# Create a KNN regressor object
k=5
knn = KNeighborsRegressor(k)

# Fit the model to the training data
knn.fit(X_train, y_train)


# Predict the house prices for the testing data
y_pred = knn.predict(X_test)

inspection=pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
inspection.head()
```

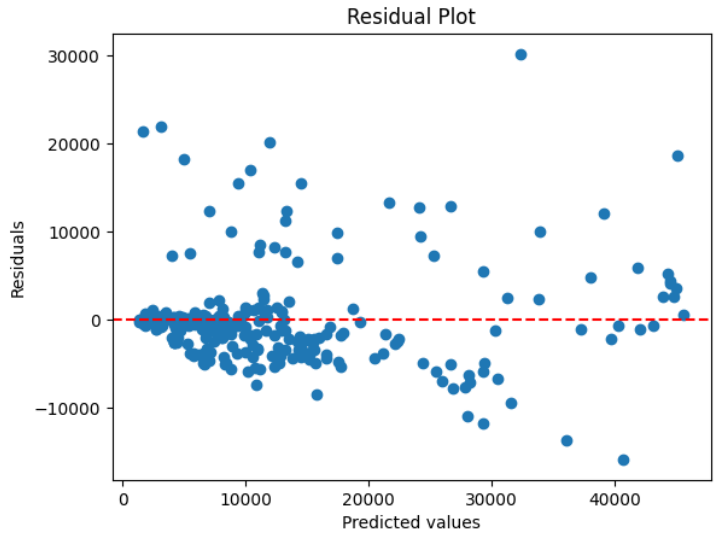
	Actual	Predicted
748	8556.90700	8490.840800
745	9910.35985	13007.466250
57	34303.16720	33060.069840
546	3268.84665	7279.783470
279	9855.13140	12867.763276

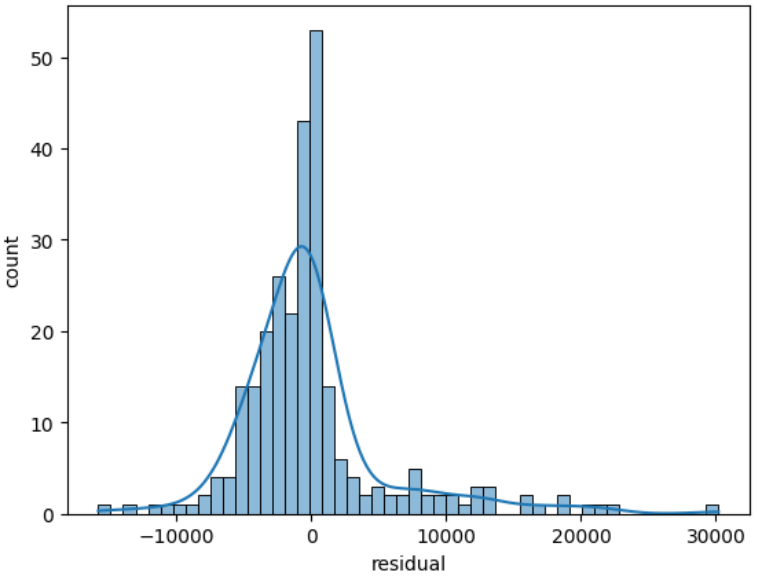
Evaluation

```
# Calculate performance metrics for the first model
rmse_1 = np.sqrt(mean_squared_error(y_test, y_pred))
r2_1 = r2_score(y_test, y_pred)
mae_1 = mean_absolute_error(y_test, y_pred)
```

```
R Squared: 0.79
Root Mean Squared Error: 5586
Mean Absolute Error: 3423
```

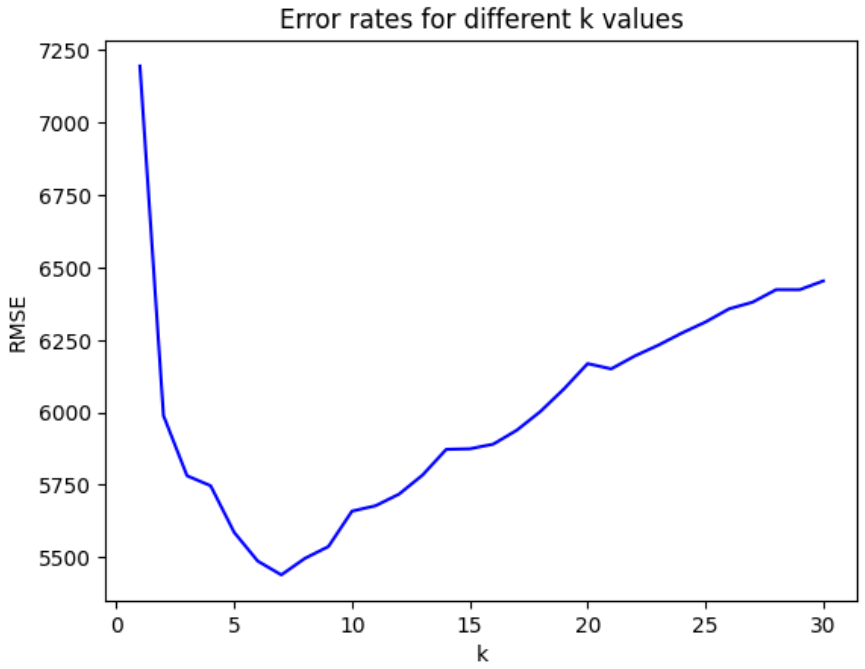
mean 13,270
std 12,110
min 1,121
max 63,770





```python
# Define a list of k values to test
k_values = list(range(1, 41))

# Train and evaluate KNN classifiers with different k values
best_k=8
best_rmse=5977
error_rates=[]
for k in k_values:
  knn = KNeighborsRegressor(k, p=2)
  knn.fit(X_train_scaled, y_train)
  y_pred_k=knn.predict(X_test_scaled)
  rmse = np.sqrt(mean_squared_error(y_test, y_pred_k))
  error_rates.append(rmse)
  if rmse <= best_rmse:
    best_k = k
    best_rmse = rmse

# Find the best k value with highest accuracy score
print(f"Best k value: {best_k}")
print(f"Best rmse: {best_rmse:.0f}")
```

```
Best k value: 5
Best rmse: 5400
```



Pros and cons

Pros

- Intuitive, simple
- No assumption about data
- Multi-class capable (majority vote)
- Can apply to regression and classification

Cons

- Slow and memory-intensive with large datasets
- Curse of dimensionality: not suitable for high dimensional data
- Sensitive to the choice of k outliers, and the shape of the data
- Struggles with imbalanced classes

Strategies to Improve Machine Learning Models (1)

Feature selection (relevant variables)

- **Remove Irrelevant Features**: e.g. recursive feature elimination, or relationship analysis to select the most relevant features.
- **Dimensionality Reduction**: e.g. Principal Component Analysis to reduce the number of features

Data pre-processing:

- **Handle Missing Values:** Fill missing values using mean, median, mode, or more advanced methods like KNN imputation.
- **Outliers**: Use techniques like z-scores, IQR (Interquartile Range), to detect outliers.
- **Normalize Data:** Standardization and Min-Max Scaling

Strategies to Improve Machine Learning Models (2)

Have more data and/or Resampling

- **Collect more data**
- **Resampling:** e.g. SMOTE (Synthetic Minority Over-sampling Technique) to handle class imbalance.

Try Different Algorithms: Experiment with various algorithms to find the best performing model.

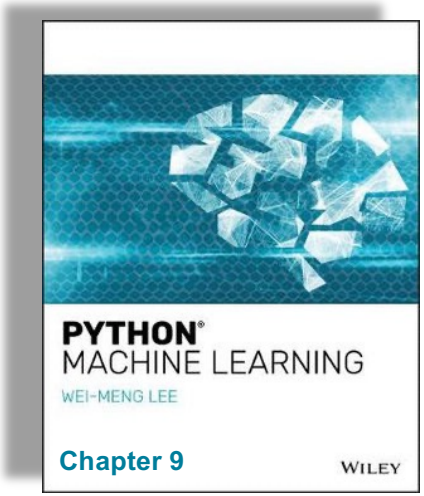
Ensemble methods: Combine multiple models to improve performance (Topic 8)

Hyperparameter Tuning: e.g. Grid Search exploring a grid of hyperparameters and select the best combination using cross-validation. (Topic 8)

KNN - Recap

- Underlaying ideas - “birds of a feather flock together”
- Can apply to binary and multi-class
- Can apply to classification as well as estimation
- Optimisation of k

From your book



https://ebookcentral-proquest-com.ezproxyb.deakin.edu.au/lib/deakin/detail.action?docID=5747364

AND See useful sites in Lab 6.

Classifiers in medical research

Zheng *et al. Visual Computing for Industry, Biomedicine, and Art* (2019) 2:17
https://doi.org/10.1186/s42492-019-0026-5

Visual Computing for Industry, Biomedicine, and Art

ORIGINAL ARTICLE Open Access

Developing global image feature analysis models to predict cancer risk and prognosis

Bin Zheng*, Yuchen Qiu, Faranak Aghaei, Seyedehnafiseh Mirniaharikandehei, Morteza Heidari and Gopichandh Danala

- Selection of features (variables) matters
- Model evaluation

Table 2 Comparison of three confusion matrices generated by three machine learning models trained using tumor-related features (Model 1), emphysema-related features (Model 2), and combined image features (Model 3) to predict the risk of lung cancer recurrence

Cancer recurrence	Model 1		Model 2		Model 3	
	Yes	No	Yes	No	Yes	No
Prediction – yes	13	14	15	19	18	10
Prediction – no	13	67	11	62	8	71

http://tiny.cc/uwqjsz