## **TEORI**

## 2.1 Klassificeringsmodeller

Klassificeringsproblem går ut på att placera data i en av flera fördefinierade kategorier. Modeller som används går ut på att generalisera och korrekt klassificera ny, osedd data baserat på relationer och mönster som identifierats i träningsdatan under träningsfasen. Det finns binära klassificerare som skiljer mellan två klasser, samt multinomialklassificerare som skilja mellan fler än två klasser. (Géron, 2019, s. 100).

Logistisk regression

Logistisk regression är en metod särskilt lämplig för att hantera binära klassificeringsproblem där målet är att förutsäga sannolikheten för att en observation tillhör en viss kategori (oftast betecknad som 0 eller 1). Kan tillämpas inom t.ex. forskning om kundbeteenden för att bedöma sannolikheten för att en kund svarar positivt eller negativt på en kampanj. (James, m.fl., 2021, s. 133).

Naive Bayes

Naive Bayes utgår ifrån att variablerna är oberoende av varandra och att alla variablerna bidrar lika mycket till resultatet. Detta antagande stämmer inte för de flesta verkliga situationer, men det förenklar de numeriska beräkningarna och gör problemet lättare hanteringsbart. (Shukla, 2024).

Support Vector Machines (SVM)

SVM är lämpad för att hantera klassificeringsproblem, då dess mål är att separera olika klasser så precist som möjligt, genom att hitta en linje eller hyperplan som har den minsta möjliga avvikelsen från datapunkterna. Med SVM har man viss flexibilitet att definiera hur mycket fel som är acceptabelt i modellen, genom att justera parametern C. (Bajallan, 2022, s. 10).

K-Nearest Neighbors (KNN)

Med KNN lagrar man alla tillgängliga värden och klassificerar varje ny data baserat på dess likhet med datapunkten som ligger närmast. Tanken bakom modellen är att punkter som ligger nära varandra brukar tillhöra samma kategori. (Bajallan, 2022, s. 16).

Beslutsträd (Decision Tree)

Beslutsträd är en övervakad algoritm som bygger strukturer som liknar träd, med s.k. beslutsnoder och lövnoder, där varje nod förgrenas till ett antal nya noder beroende på antalet möjliga utfall. De noder som inte själv förgrenas kallas för ”löv”. Beslutsträd är en serie beslut som tas utifrån given data i en bestämd följdordning för att nå ett specifikt resultat. Medan algoritmen är ganska lätt att använda, är ett enda träd oftast inte tillräckligt för att ge effektiva resultat. (Sharma, 2024).

Random Forest Classifier (RFC)

RFC kombinerar förutsägelserna från flera individuella beslutsträd för att generera det slutliga resultatet. Detta är en slumpartad teknik som kombinerar resultatet av flera individuella modeller och fattar det slutliga beslutet baserat på majoriteten av utfallen. RFC kan på så vis förbättra modellens övergripande prestanda och robusthet jämfört med enskilda beslutsträd. (Sharma, 2024).

## Modellutvärdering

Confusion Matrix

Förvirringsmatrisen (confusion matrix) är en tabell som används för att utvärdera prestandan för en klassificeringsmodell, genom att tillhandahålla en sammanfattning av de förutsägelser som modellen gör. Förvirringsmatrisen är en tabell som visar fyra värden:

• True positive (TP): Datapunkter som vi förutspådde positiva som faktiskt är positiva.  
• False positive (FP): Datapunkter som vi förutspådde positiva som faktiskt är negativa.  
• False negative (FN): Datapunkter som vi förutspådde negativa som faktiskt är positiva.

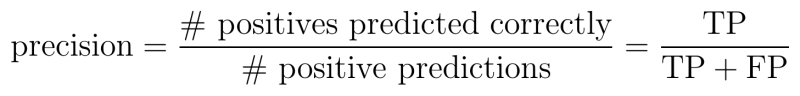
• True negative (TN): Datapunkter som vi förutspådde negativa som faktiskt är negativa.  
  
Den första och sista är de datapunkter som förutspåddes korrekt och den andra och tredje är de datapunkter som förutspåddes felaktigt. Vi kan använda de fyra värdena i förvirringsmatrisen (TP, TN, FP, FN) för att beräkna olika mätvärden som noggrannhet, precision, träffsäkerhet och F-mått.

(Narkhede, 2018)

Mätvärden

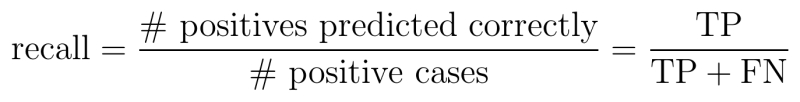
Noggrannhet (Accuracy) avser procentandelen förutsägelser som är korrekta. Om vi t.ex. har 100 datapunkter och förutsäger 70 av dem korrekt är noggrannheten 70/100 = 0.7 = 70 %. Noggrannhet är ett bra mått om våra klasser är jämnt fördelade, men kan vara missvisande vid obalans mellan klasserna. (Géron, 2019, s. 90). Därför kan det med fördel kompletteras med precision, recall och F score för en mer detaljerad bild.

Precision är procentandelen av modellens positiva förutsägelser som är korrekta. (Narkhede, 2018)



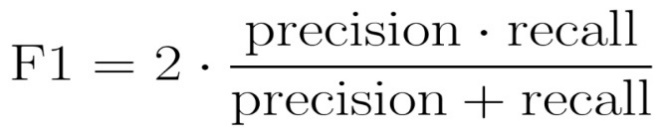
Figur 1. Precision = Av alla datapunkter som vi förutspått som positiva, så här många är faktiskt positiva.

Träffsäkerhet (Recall)är andelen positiva fall som modellen förutsäger korrekt. (Narkhede, 2018)



Figur 2. Träffsäkerhet = Av alla datapunkter med positivt värde, så här många förutspåddes som positiva (alltså korrekt).

F1-mått (F1 score) är ett genomsnittsvärde av precision och träffsäkerhet, en kompromiss när vi inte kan välja mellan två modeller då den ena har högre precision och den andra högre träffsäkerhet. (Géron, 2019, s. 92).



Figur 4. Den matematiska definitionen av F1-måttet där man multiplicerar produkten av precisionen och träffsäkerheten

med 2, och delar resultatet med summan av precisionen och träffsäkerheten.

## **METOD**

Syftet har varit att förutsäga värdet på saknade instanser i kolumnen "Period/Datering" baserat på värdena i motsvarande instanser i kolumnerna "Kommun", "Koordinater", "Materialtyp" och "Föremål". Uppgiften bedömdes som ett klassificeringsproblem, eftersom vi vill förutsäga ett kategoriskt utfall (värden i variabeln " Period/Datering") baserat på värdena i andra variabler. Med tanke på dettaskulle Logistic Regression eller Support Vector Classifier (SVC) vara lämpliga modeller.

Första steget har varit att förbereda data och se till att de olika variablerna konverteras till lämpligt string-, numeriskt-, eller tidsformat. Vi använde oss av två olika metoder för att tvätta datan. Den ena går ut på att dela in all data i kategorier, i vårt falla omvandla årtal och kortare perioder till kategorier som omfattar större tidsperioder (Bilaga - - - - -), vilket resulterade i 20 kategorier (15 kända och 5 okända) som vardera innehåller de årtalen som ryms inom den tidsperioden).

Den andra metoden för att tvätta datan var genom Regex (Regular Expression) som är ett verktyg för att matcha textmönster och extrahera information från kod, loggfiler och andra texter. Regex-mönstret som använts \d{3,4}-\d{3,4} matchar ett intervall med år (t.ex. 1100-1150 eller 100-200). Funktionen clean\_date extraherar den första förekomsten av det här mönstret och ignorerar alla andra tecken (kommentarer, bokstäver osv.). Detta tillämpades på hela kolumnen och de rensade resultaten lagrades i en ny kolumn.

Det visade sig att regex-mönstret inte matchade datan fullt ut. Mönstret \d{3,4}-\d{3,4} letar specifikt efter sekvenser som 1100-1200. Databasen innehåller dock andra format som "900-t" eller "efter 1050" som inte matchar det här mönstret. För att göra funktionen mer flexibel uppdaterades regex för att ta hänsyn till en större mängd numeriska mönster, till exempel:

* + - Enstaka år (t.ex. 900)
    - Årsintervall (t.ex. 1100-1150)
    - År med bokstäver eller andra symboler efter (t.ex. 900-t eller 1100-t)

Funktionen uppdaterades med ett mer omfattande regex:

python

# Updated function to clean the 'Period\_Datering' column

def clean\_date(date\_string):

if isinstance(date\_string, str): # Check if the value is a string

# Extract the first sequence of 3 or 4 digits, possibly followed by a dash and another sequence

clean\_date = re.search(r'\d{3,4}(-\d{3,4})?', date\_string)

if clean\_date:

return clean\_date.group(0) # Return the cleaned date (e.g., '1100-1150' or '900')

return None # If no valid date format is found, return None

# Applying the cleaning function to the 'Period\_Datering' column

df['Cleaned\_Period\_Datering'] = df['Period\_Datering'].apply(clean\_date)

Det nya regex-mönstret \d{3,4}(-\d{3,4})? matchar både enskilda år (t.ex. 1100) och årsintervall (t.ex. 1100–1150). Den kunde nu samla in årsformat med ett potentiellt intervall och ignorera bokstäver

eller symboler efter åren. På så sätt kunde årsdata samlas in korrekt även när formatet varierat.

Efter att datan tvättats ville vi prediktera perioden för de instanser där värden saknas i databasen, med utgångspunkt i de instanser där värden finns. Eftersom vi inte har några facitsvar på när de runstenarna, där period ej finns angiven, byggdes har vi valt att jämföra prediktionerna som gjorts med de två olika metoderna att tvätta data och se huruvida prediktionerna överensstämmer med varandra samt i vilken grad. Ett ytterligare steg för att dubbelkolla att våra prediktioner stämmer har varit att jämföra de predikterade perioderna med de perioder som olika ristare varit verksamma och skapat runstenarna (informationen från kolumnen ”Ristare” i tabellen).

Vi ville jämföra noggrannheten och andra mått (precision, träffsäkerhet, F1-poäng) från klassificeringsrapporterna för båda modellerna för att kunna välja den modell som ger bäst prestanda. Datan delades upp med train\_test\_split metoden för att skapa datauppsättningar för träning och testning. Träning av modellen har gjorts med Logistic Regression, SVC och RandomForestClassifier. Utvärdering av modellen på testdata gjordes med hjälp av noggrannhet, precision, återkallelse, etc.

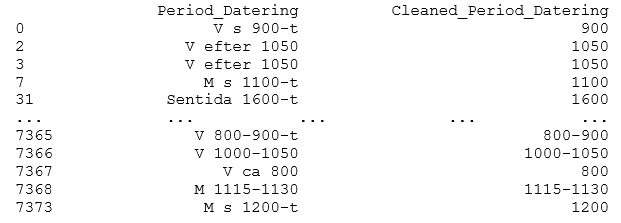
Utmaningen kom med varningen “UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples”, vilket inträffar när modellen inte förutsäger några instanser av vissa klasser (etiketter). Detta kan inträffa vid klassificeringsproblem där vissa klasser är underrepresenterade (eller inte representerade alls) i träningsuppsättningen eller testuppsättningen. Modellen kan inte klassificera vissa kategorier korrekt och förutsäger bara en delmängd av klasserna.

Varningen åtgärdades på så vis att det gjordes kontroll av klassfördelningen, om den är skev (dvs. att vissa klasser har mycket få eller inga förekomster), samt ställde in zero\_division parametern. Det visade sig att t.ex. kolumnen ”Föremål” har en mycket obalanserad fördelning, där vissa klasser har många instanser (t.ex. klass 785 har 172 prover) och många andra har mycket få instanser (t.ex. flera klasser har bara 1 prov). Den här obalansen gjorde troligen att din modellerna presterade dåligt i underrepresenterade klasser. Samma kontroll gjordes för de övriga variablerna.

Vid nya testningar användes modeller med klassviktning (t.ex. model = RandomForestClassifier(class\_weight='balanced', random\_state=42) för att automatiskt justera vikter baserat på klassfördelningen. Detta lyckades dock inte råda bot på obalansen i fördelningen av de olika klasserna som i sin tur påverkat prediktionerna. Detta ser vi när vi jämför de predikterade perioderna med de perioder som olika ristare varit verksamma, vilket presenteras i nästa avsnitt.

# **Resultat och Diskussion**

Detta är de nya värdena, avseende Period/Datering efter att datan tvättats med Regex:

****

Samtliga värden (text, bokstäver, symboler, mm) förutom de numeriska har tagits bort, vilket gör att värden som avser samma tidsperiod, men skrivits på olika sätt, blir identifierade som likadana. Resultatet blev att de unika värdena minskat från 587 till 241.

Nedan ser vi de unika värdena i kolumnen "Period/Datering" efter att datan indelats i kategorier:

numeric\_values = [

'folkvandringstid 160–375',

'högmedeltid 1150–1300',

'mellanvendeltid 600–700',

'modern tid 1500–1700',

'modern tid 1700–1800',

'modern tid 1800–1900',

'sen vikingatid 1050-1200',

'senmedeltid 1300–1500',

'sent folkvandringstid 375–500',

'sent vendeltid 700–800',

'tidig medeltid 1000–1150',

'tidig vendeltid 500–600',

'tidig vikingatiden 700-800',

'vikingatiden 800-1050',

'vikingatiden 800–1050'

]

non\_numeric\_values = [

'allmän/osäker sentida',

'oklar/allmän vendeltid',

'okänd',

'osäker/allmän medeltida',

'unknown'

]

Så här såg värdena ut efter att prediktionerna gjorts (exemplet visar data som tvättats med första metoden, dvs. blivit indelad i kategorier):

Period/Datering Predicted\_Period/Datering

0 allmän/osäker sentida vikingatiden 800–1050

1 allmän/osäker sentida senmedeltid 1300–1500

2 allmän/osäker sentida vikingatiden 800-1050

3 allmän/osäker sentida vikingatiden 800-1050

4 allmän/osäker sentida vikingatiden 800-1050

... ... ... ... ...

11666 vikingatiden 800–1050 NaN

11667 vikingatiden 800–1050 NaN

11668 vikingatiden 800–1050 NaN

11669 vikingatiden 800–1050 NaN

11670 vikingatiden 800–1050 NaN

Rader i tabellen som saknar ett numeriskt värde för kolumnen ”Period/Datering” har ersatts med predikterade värden. För raderna där det finns ett numeriskt värde har givetvis inga prediktioner gjorts, utan dessa rader har utgjort träningsdatan.

Så här ser resultatet ut med modellen som visats sig bäst (RFC):

Totala matchningar: 6435

Korrekt matchningar (True Positives): 5745

Felaktiga matchningar (False Positives): 690

Felaktiga icke-matchningar (False Negatives): 122

Accuracy: 89.28%

Precision: 89.28%

Recall: 97.92%

F1 Score: 93.40%

**Enligt resultaten vi fått är det totala antalet matchningar 6435, dvs. d**et totala antalet gånger då en matchning överhuvudtaget identifierades mellan värdena i de två kolumnerna. Av de totala matchningarna är 5745 korrekta **(True Positives)**, dvs. rader där värdena från de två kolumnerna överensstämmer med de förväntade perioderna enligt indelningen. **Felaktiga matchningar**: Det finns 690 felaktiga matchningar **(False Positives),** dvs. rader där data felaktigt har identifierats som en matchning men där värdena egentligen inte överensstämmer. Antalet rader där en verklig matchning inte identifierades **(False Negatives)** är 122. Dessa är alltså "missade" korrekta matchningar.

**Värdet på noggrannheten (Accuracy) ligger på 89.28%, vilket betyder att nästan 90 procent** av det totala antalet matchningsförsök varit korrekta, dvs. Metoden har alltså haft en hög noggrannhet, men även en lika hög precision, vilket betyder ett fåtal felaktiga matchningar. Recall mäter hur många av de faktiska matchningarna som identifierades korrekt. Ett värde på nästan 98% avseende träffsäkerheten (Recall) betyder att metoden är mycket känslig för att upptäcka faktiska matchningar, även om det innebär att en del felaktiga identifieras som matchningar. F1-värdet är en balanserad kombination av precision och recall och indikerar metodens generella tillförlitlighet. Här är värdet 93.4%, vilket innebär att metoden överlag är mycket pålitlig för att upptäcka korrekta matchningar och undvika felaktiga.

OBS! När vi jämförde våra predikterade perioder med perioderna då enskilda ristare varit verksamma visade det sig att modellen presterar mycket bra på majoritetsklasserna (d.v.s. ristarna som förekommer i många instanser) men kämpar med minoritetsklasserna (d.v.s. ristarna som förekommer i få, ofta bara 1 instans).

## **SLUTSATSER**

Sammanfattningsvis visar resultaten att vår metod är **mycket effektiv** för att identifiera rätt matchningar, med hög recall och precision, vilket ger ett balanserat och högt F1-score. Det betyder att metoden är bra på att både upptäcka matchningar och minska falska matchningar, vilket ökar trovärdigheten och pålitligheten i studien. För att förbättra resultatet ytterligare kan man försöka minska antalet **false positives** och **false negatives,** exempelvis genom att finjustera periodmappningen och överväga fler eller stramare kriterier för matchningarna. Detta för att försöka fånga upp även minoritetsklasserna.