# Sifferigenkänning

## baserad på MNIST-Datasetet


Kawser Ayoub

EC Utbildning

Maskininlärning

202403

# Abstract

In this rapport, a collection of handwritten digits called MNIST was explored. The process involved data fetching, exploratory analysis and preprocessing techniques to prepare the dataset for machine learning applications. Three machine learning models, Logistic Regression, Random Forest and Support Vector Machine were applied to estimate their effectiveness in recognizing handwritten digits. The result was that Random Forest was the most effective, achieving a test accuracy of 96.91%. Along the way, the practical application of the model was tested through digit prediction from real world images with reasonable accuracy, although some misclassifications were noted.

# Innehållsförteckning

# 1. Inledning

I en värld där penna möter papper, uppstår en dans av linjer och kurvor som vi känner igen som siffror. Men när dessa handgjorda tecken ska tolkas av maskiner, hur väl förstår de egentligen vår handstil? MNIST är en förkortning för Modified National Institute of Standards and Technology, en databas som erbjuder en portal till denna världen med sin omfattande kollektion av 70 000 bilder och representerar en bred variation av stilar från olika individer.

Syftet med denna rapport är att utvärdera och jämföra prestandan hos olika maskininkärningsmodeller inom handskriven siffer igenkänning med hjälp av MNIST databasen. För att uppfylla syftet kommer följande två frågeställningar att utforskas:

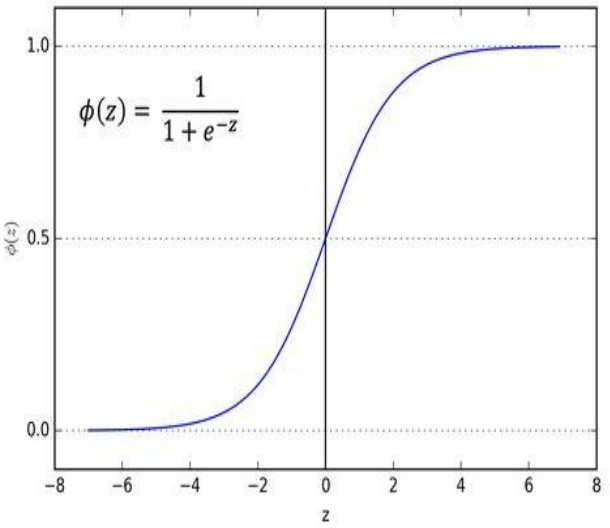
1. Kan modellerna uppnå och överträffa en noggrannhet nivå på 80%?

2. Hur presterar den bästa modellen vid klassificering av verkliga handskrivna siffror och vilka är de vanligaste misstagen den gör?

# 2. Teori

## 2.1 Logistisk Regression

Logistisk Regression som beskrivs av IBM är en statistisk modell som ofta används för klassificering och prediktiv analys. Den uppskattar sannolikheten  för att en händelse inträffar, t.ex. som att rösta eller inte, baserat på en uppsättning oberoende variabler. Modellen använder en logistisk funktion för att transformera oddsen (sannolikhet för framgång jämfört med sannolikheten för misslyckande) till ett sannolikhetsmått. Parametrar i modellen, representerade av beta-koefficienter, uppskattas vanligtvis via maximal sannolikhet uppskattning för att hitta den bästa passformen. Denna modell är särskilt användbar i binär klassificering, där utfall förutspås som 0 eller 1 baserat på sannolikhet.

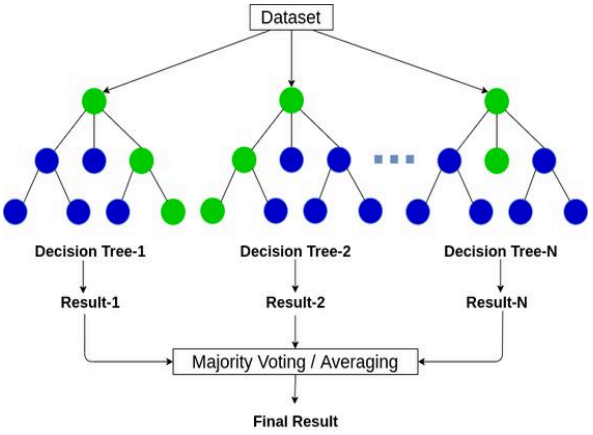



*Figur 1. Logistisk Regression algoritm*

I projektet tillämpas logistisk regression på MNIST datasetet med hjälp av sklearns LogisticRegression funktionen inom en förbearbetningspipeline som inkluderar StandardScaler. Detta tillvägagångssätt standardiserar bilddata, vilket underlättar mer exakta modell förutsägelser om vilken siffra varje bild representerar. Vidare utvärderas modellens prestanda genom validering noggrannhet, vilket informerar om hur väl den logistiska regressionsmodellen skiljer mellan olika handskrivna siffror baserat på deras pixelvärden.

## 2.2 Random Forest

Random Forest som förklarats av IBM är en maskininlärning algoritm utvecklad av Leo Breiman och Adele Cutler. Det är känt för att kombinera flera beslutsträd för att producera en mer exakt och stabil förutsägelse genom att beräkna ett genomsnitt av resultaten. Denna ensemble metod fungerar för både klassificerings och regressions problem, vilket förbättrar prestandan genom att minska risken för överanpassning i samband med enstaka besluträd. Tekniken involverar packning (bootstrap aggregation) och har slumpmässighet för att säkerställa låg korrelation mellan träden, vilket förbättrar den övergripande prediktions noggrannheten.

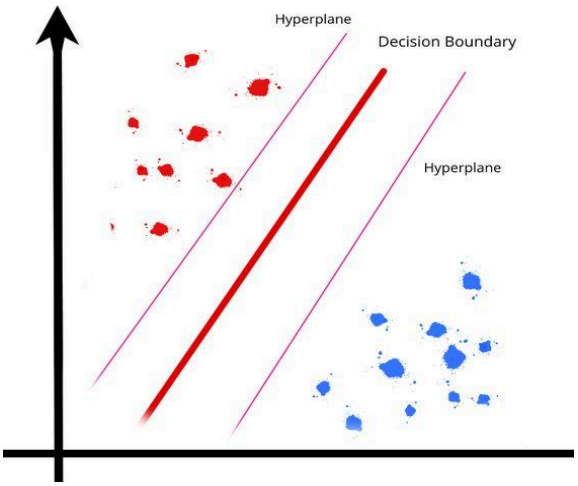



*Figur 2. Random Forest algoritm*

I projektet används Random Forest algoritmen via sk learns Random Forest Classifier. Denna metod använder flera beslutsträd för att förutsäga klassen av handskrivna siffror från MNIST datasetet, vilket förbättrar prediktions noggrannheten och robustheten jämfört med att använda ett enda beslutsträd. Ensemble-metoden mildrar överanpassning och förbättrar generaliseringen över osynlig data.

## 2.3 Support Vector Machine

Support Vector Machines (SVM) enligt IBM är övervakade inlärningsmodeller som främst används för klassificeringsproblem. SVM:er utvecklades på 1990-talet och fungerar genom att hitta ett optimalt hyperplan som separerar datapunkter i klasser i ett N-dimensionellt utrymme. Metoden maximerar marginalen mellan de närmaste punkterna i olika klasser, så kallade stöd-vektorer, för att förbättra modell-generaliseringen för ny data. SVM:er hanterar både linjära och olinjära data, med hjälp av kärnfunktioner som linjär, polynom och radiell basfunktion (RBF).




*Figur 3. SVM algoritm*

I projektet appliceras SVM modellen med en RBF kärna direkt på MNIST datasetet för siffer-klassificering. Denna modell är en del av en pipeline som inkluderar Standard Scaler för data normalisering. Skalnings steget säkerställer att SVM arbetar på data som har skalade funktioner vilket är fördelaktigt för den gradient nedstignings-process som är inneboende i SVM optimering.

## 2.4 Grid Search CV

I artikeln Hyperparameter Tuning with GridSearchCV skriven av Great Learning Team beskriver de att Grid Search CV är en metod som används inom maskininlärning för att optimera parametrarna för en modell för bästa prestanda. Genom att definiera ett rutnät av parametrar att testa, arbetar den systematiskt genom flera kombinationer och kors validerar allt för att bestämma vilken inställning som ger bäst resultat baserat på en specificerad poäng. Denna metod är effektiv för att förbättra modellens noggrannhet och säkerställa att modellen generaliserar väl på ny data.

I projektet användes Grid Search CV på modellen Random Forest. Hyperparametrar som används är:

```
rf_gs_param = {
     'rf__n_estimators': [50, 100],
    'rf__max_depth': [None, 10, 20, 30],
    'rf__min_samples_split': [2, 5, 10],
    'rf__min_samples_leaf': [1, 2, 4]
}

rf_gs = GridSearchCV(rf_pipeline, rf_gs_param, cv=3, verbose=2, n_jobs=-1)
rf_gs.fit(X_train, y_train)
```

*Figur 4 - GridSearchCV kod på Random Forest*

**n_estimators:** Varierad för att observera effekten av att öka antalet träd på modellens prestanda. Fler träd kan leda till bättre noggrannhet men också öka beräknings-komplexiteten.

**max_depth:** Testad med olika djup för att hitta det optimala djupet som förhindrar överanpassning. Att begränsa djupet kan hjälpa till att minska modellens komplexitet och förbättra dess generalisering.

**min_samples_split:** Denna parameter påverkar det minsta antal urval som krävs för att dela en skärningspunkt. Att justera detta värde hjälper till att kontrollera trädets tillväxt och att förhindra överanpassning genom att se till att delningar bara inträffar när det finns tillräckligt med prover för att fatta ett statiskt beslut.

**min_samples_leaf:** På samma sätt som min_samples_split säkerställer den här parameter att varje blad har ett minsta antal urval, vilket hjälper att undvika alltför komplexa modeller som över anpassar tränings-datan,

## 2.5 Confusion Matrix

En confusion matrix är en tabell som används i klassificerings-uppgifter för att visualisera prestandan hos en algoritm. Den kontrasterar de faktiska kontra förutspådda klassificeringarna för att ge detaljerad inblick i en modells noggrannhet. Vanligtvis ett 2x2 rutnät inkluderar det sanna positiva (TP), där modellen korrekt förutsäger den positiva klassen; sanna negativ (TN), där den korrekt förutsäger den negativa klassen; falska positiva (FP), där det felaktigt markerar ett negativt som positivt (TYp 1-fel); och falskt negativ (FN), där det missar ett positivt fall (Typ II-fel). Från dessa värden kan mätvärden som noggrannhet, precision, återkallelse och F1-poäng härledas, vilket ger en nyanserad bild av modellens prestanda.

# 3. Metod

## 3.1 Data fetching

Datan hämtades från Modified Nation Institute of Standards and Technology (MNIST), en allmänt erkänd resurs inom området maskininlärning för handskriven sifferigenkänning. MNIST nåddes via funktionen ‘fetch_openml’ från sickit-learn.biblioteket. Datan importerades till Python miljön Jupyter Notebook med bland annat bibliotek som numpy för numeriska operationer, pandas för datamanipulation, matplotlib för datavisualisering, joblib för att spara och ladda modeller och OpenCv(cv2) för bildbehandling. Specifikt hämtades “mnist_784”,

Vid hämtning delades datasetet upp i arrays: ‘X’ som innehåller bilddata och ‘y’ som innehåller motsvarande etiketter för siffrorna. För att säkerställa kompatibilitet med klassificerings-algoritmer och konsistens i datatyp, konverterades mål etiketterna ‘y’ till 8 bitars heltal utan tecken (np.uint8).

## 3.2 EDA

För en djupare förståelse utforskades datan genom att inledningsvis undersöka strukturen och komponenterna i datamängden med ‘.keys()’-metoden, vilken avslöjade attribut som ‘data’, ‘target’ och ‘DESCR’, bland annat. Attribut som inkluderades ifrån ‘mnist.keys()’ är: ‘mnist.url’, och ‘mnist.DESCR’. Sedan användes 'shape'-funktionen för att förstå indelningen av ‘X’ och ‘y’. För att säkerställa transformationen av ‘np.unit8’ skrevs datatyperna ‘X.dtype’ och ‘y.dtype’ ut. Vidare utforskades de tio första bilderna tillsammans med deras motsvarande etiketter för att illustrera variationen och egenskaperna hos de handskrivna siffrorna. Slutligen verifierades databasens integritet genom att kontrollera saknade värde som hjälpte att avslöja om datan var komplett och lämplig för vidare bearbetning.

## 3.3 Träning, validering, finjustering och testing

Datan var uppdelad i tre delmängder: träning, validering och test. De första 50 000 instanserna tilldelades till träning (X_train, y_train), de efterföljande 10 000 instanserna till validering (X_val, y_val) och de sista 10 000 instanserna till testsetet (X_test, y_test). För modell-träning valdes Logistic Regression, Random Forest och Support Vector Machine. För att effektivisera bearbetningen och modelleringsprocessen konstruerades pipelines för varje modell med hjälp av scikit-learn - Pipeline. Varje pipeline bestod av en Standard Scaler för funktions-skalning följt av respektive klassificerare.

Valideringsprocessen genomfördes separat för var och en av de tre modellerna. Alla tre modeller utvärderades genom att använda ‘.predict()’ metoden på validerings-setet (X_val). Efter detta beräknades validerings noggrannheten genom att jämföra de förutsagda etiketterna mot de faktiska etiketterna (y_val) med hjälp av metoden accuracy score från modulen

sklerarn.metrics. Slutligen definierades en funktion med namnet display_confusion_matrix för att plotta förvirrings-matrisen.

Efter resultatet från validerings testet användes Grid Search CV för att finjustera Random Forests prestanda. De valda parametrarna är n_estimators, max_depth, min_samples_split och min_samples_leaf. Målet var att förbättra Random Forest modellens noggrannhet och generalisering av osedd data samt balansera mellan under-anpassning och  överanpassning.

Slutligen extraherades den bäst presterande modellen från Grid Search CV, betecknad som “best_rf_model”. Denna modellen tillämpas sedan på test setet ‘X_test’ för att förutsäga motsvarande etiketter, vilket resulterar ‘y_test_pred_rf’. Noggrannheten för dessa förutsägelser i jämförelse med de sanna etiketterna ‘y_test’ beräknas med hjälp av funktionen accuracy score, vilket ger ett kvantitativt mått på modellens prestanda på osedda data. Sedan genereras en confusion matrix och slutligen sparas den bästa estimator från grid search, ‘best_rf_model’, lokalt som en pickle fil med hjälp av joblibs dump funktionen. Detta gör att modellen kan lagras och senare kommas åt eller distribueras utan att behöva tränas om.

### 3.4 Streamlit - siffer prediktioner

Streamlit används för att skapa en webbapplikations gränssnitt. Bibliotek som importeras för detta ändamål är  numpy, opencv och joblib. Den tränade maskininlärning modellen, ‘best_rf_model’ laddas med hjälp av joblibs laddningsfunktion. En funktion, ‘process_image’ är definierad för att bearbeta ingångs bilden till gråskala, ändrar storlek på dem till 28x28 pixlar, normaliserar pixelvärdet, inverterar färgerna, tillämpar ett tröskelvärde för att förbättra kontrasten och plattar ut bilden till en endimensionell array som är lämplig för modell inmatning.

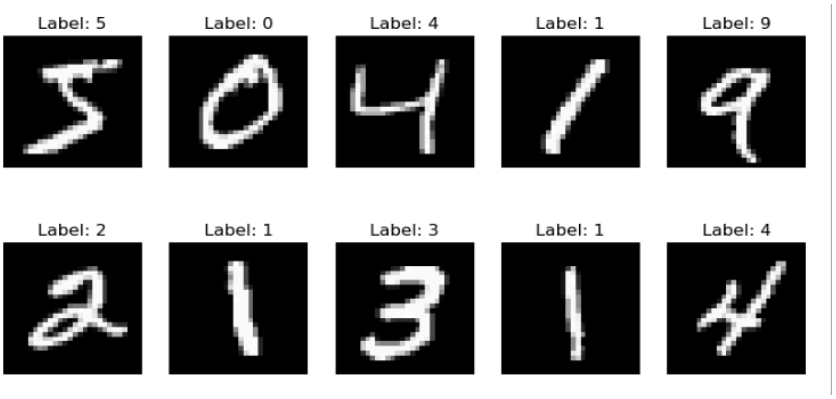
I streamlit användargränssnittet ställs en titel in samt en knapp skapas som, när den trycks ned, utförs förutsägelser på fördefinierade siffror. Varje bild bearbetas med funktionen ‘process_image’ och den bearbetade gråskala bilden visas. De tränade Random Forest modellen förutsäger sedan siffran baserat på den bearbetade bilden. Förutsägelserna visas bredvid motsvarande bild. Miljön som streamlit koden skapades i är Jupyter Lab. Det var nödvändigt att skriva koden på en integrerad utvecklingsmiljö (IDE) så som Jupyter Lab eftersom Jupyter Notebook inte är designad för det.

# 4. Resultat och Diskussion

## 4.1 EDA

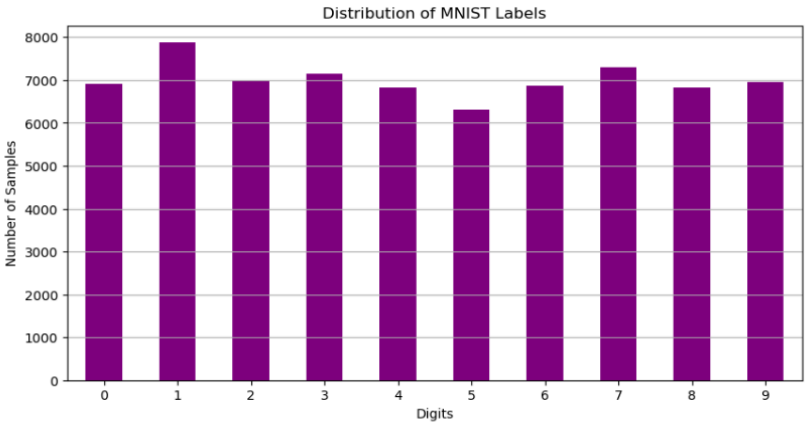
Data-uppsättningen består av  70 000 urval, var och en representerad av 784  funktioner, motsvarande pixlarna i 28x28 bilder, som matchar de förväntade dimensionerna som krävs för bildbaserade maskininlärning uppgifter. Verifieringen av datatyp avslöjade att bildfunktioner lagras som heltal (int64) och etiketterna som osignerade heltal (uint8), vilket är i linje med kraven för bilddata-manipulering och klassificerings-åtaganden.

Visuell inspektion av de första tio bilderna tillsammans med deras etiketter illustrerade datasets mångfald i handskriftsstilar, vilket underströk komplexiteten i utmaningen med sifferigenkänning. En sådan preliminär visualisering hjälper till att acklimatisera sig till datasetet, vilket främjar en nyanserad förståelse för potentiella behov för bearbetning.




*Figur 5 - Tio första bilderna*

Distributionen av MNIST etiketterna indikerade en nästan enhetlig fördelning över de tio siffriga kategorierna. Denna jämvikt minskar risken för model bias mot överrepresenterade klasser vilket säkerställer en mer robust och generaliserbar inlärningsprocess.



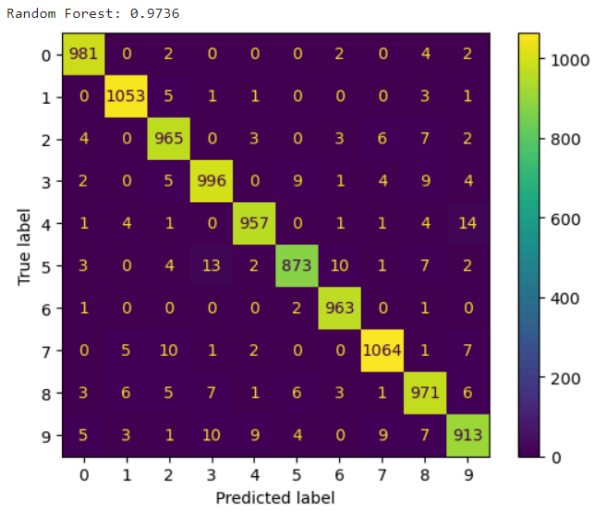

*Figur 6 - Distribution av MNSIT etiketter*

Slutligen undersöktes om det finns en närvaro av saknade värden och resultatet var det inte fanns saknades värden. Detta understryker kvaliteten på datan, vilket innebär att man kan fokusera på modellutveckling och optimering snarare än data förbehandling.

## 4.2 Träning, validering, finjustering och test

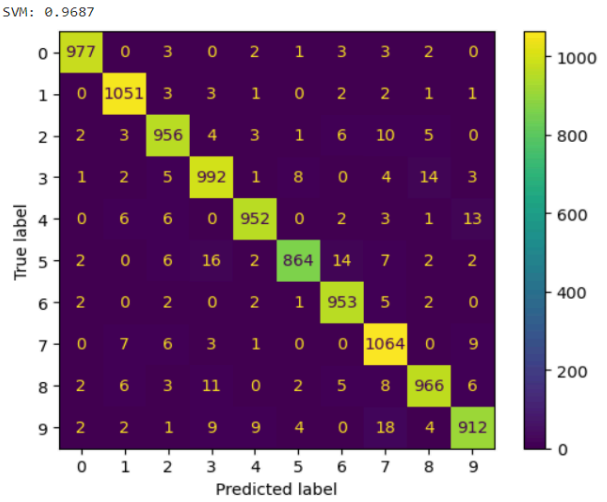
Kan modellerna uppnå och överträffa en noggrannhet nivå på 80%?

Random Forest överträffade med en noggrannhet på 97,36%. (valideringstest)



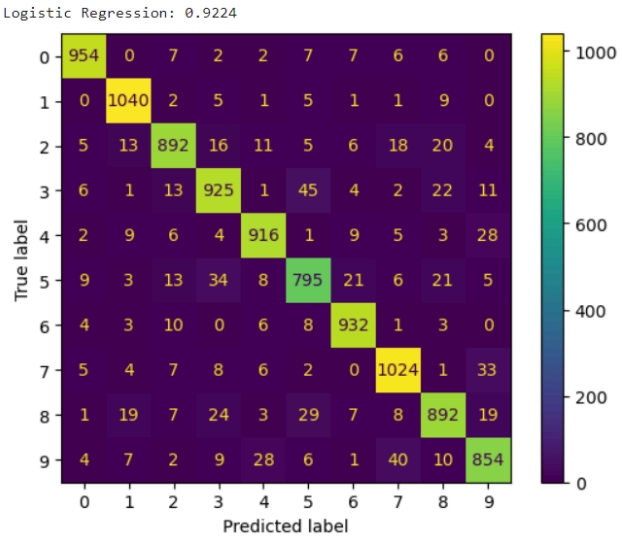

*Figur 7 - Random Forest validering*

SVM följde med en noggrannhet på 96,87%. (valideringstest)



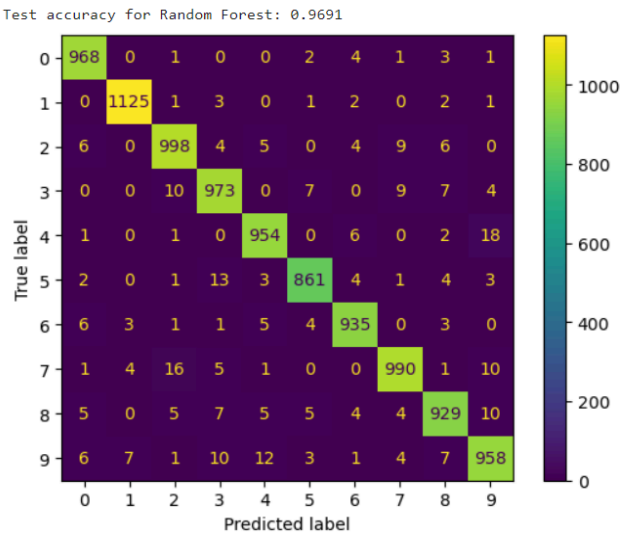

*Figur 8 - SVM validering*

Logistisk Regression nådde en noggrannhet på 92,24%. (valideringstest)




*Figur 9 - Logistisk regression validering*

Random Forest uppnådde 96,91% i noggrannhet (test-set)




*Figur 10 - Random Forest på test*

I valideringstestet presterade Random Forest bättre i noggrannhet än de två andra modeller. Med detta resultat genomfördes en finjustering på den främsta modellen genom GridSearchCV som identifierade optimala parametrar för att bäst anpassas till datasetets egenskaper. Processen pekade ut en specifik konfiguration (‘max_depth’: 30,

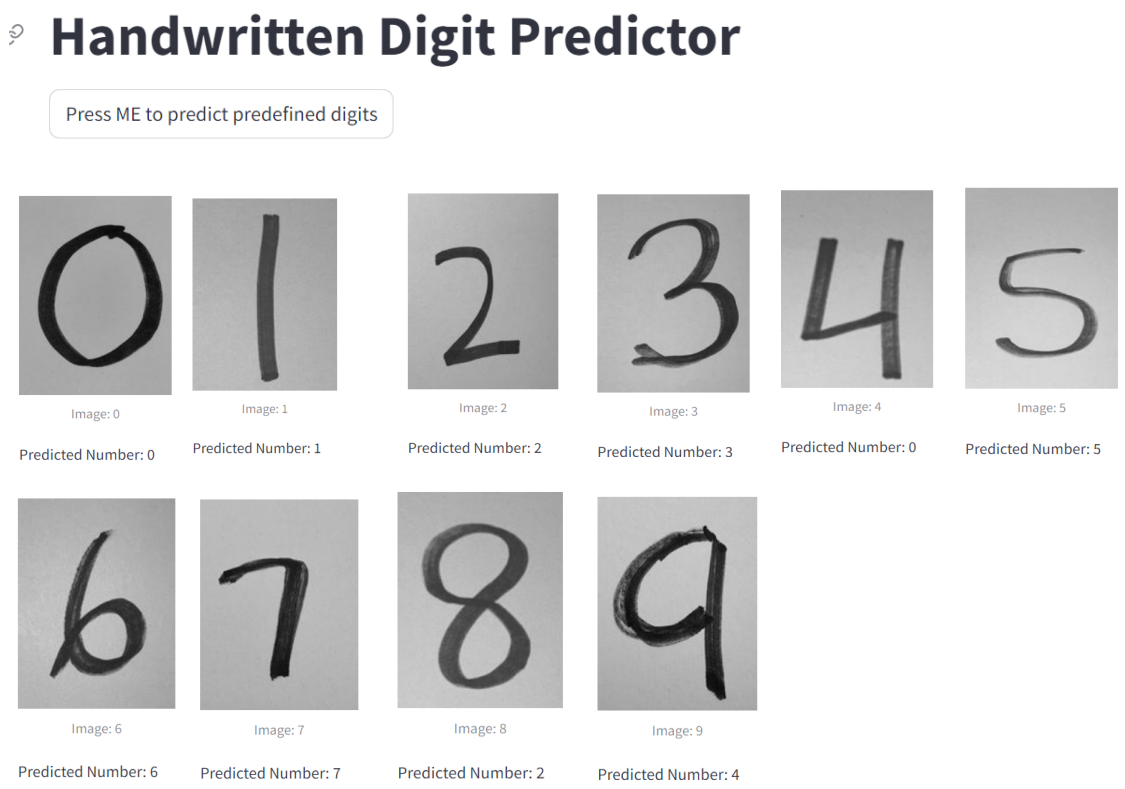
‘max_samples_leaf’:1, ‘min_samples_split’: 2, ‘n_estomators’: 100) som gav den korsvalideringspoängen (96,26%) bland de övervägda parametrarna.

Den slutliga test utvärderingen fungerade som en prövning av modellens förmåga att generalisera till nya data, vilket bekräftade dess robusthet och tillförlighet. Som tidigare nämnts uppnådde Random Forest en noggrannhet på 96,91% på testdatan.

### 4.3 Streamlit - siffer prediktioner

Hur presterar den bästa modellen vid klassificering av verkliga handskrivna siffror och vilka är de vanligaste misstagen den gör? Resultatet av streamlit koden visade sin förmåga att förutsäga siffror exakt för de flesta bilder. Förutsägelserna var som följer: siffrorna 0 till 2 och 5 till 7 identifierades korrekt. Å andra sidan observerades avvikelser i modell förutsägelser för bilder 4, 8, och 9 där de klassificerades som 0,2 respektive 4.

Detta kan möjligen bero på variationer i siffrors presentation som inte var tillräckligt representerade. Å andra sidan upptäcktes vikten av bildkvalitet, belysning, färg, bakgrund, positioner och handstil för prediktionen under bildbehandlingen. Högre upplösning och tydlighet i bilden förbättrar modellens förmåga att urskilja detaljer i handstilen. Å andra sidan bör bilden skalas ned till storleken som modellen tränas på, 28x28, vilket kan påverka detaljnivån. Siffrorna bör vara centrerade i bilden på samma sätt som i träningsexempel. Rotation, lutning eller skalningen som avviker från träningsexempel kan också försämra prediktionen. Trots att det finns 70 000 urval på handstilar kan extremt unika eller ovanliga handstilar som saknas i träningsexempel vara svåra för modellen att prediktera korrekt.



# 5. Slutsatser

Utforskningen och analysen av MNIST och det efterföljande arbetet med att träna, validera, finjustera och testa olika maskininlärnings-modeller ger en viktig kunskap om hur dessa modeller fungerar och hur de kan användas i verkliga applikation så som en Streamlit applikation. Det identifierades att datasetet är välorganiserat och innehåller data av hög kvalitet, vilket är viktigt för att kunna utveckla och förbättra maskininlärnings-modeller.

Kan modellerna uppnå och överträffa en noggrannhet nivå på 80%? Ja, alla modeller uppnådde en noggrannhet nivå över 80%. Denna process från början till slut visar hur viktigt det är med noggrannhet genom alla steg för att skapa en modell som inte bara fungerar bra i tester men också i verkliga situationer.

De fel som uppstod kan vara tecken på brist i modellerna som valdes eller processen som maskin-inlärningen genomfördes på. Å  andra sidan kan det också innebära att bildhanteringen inte genomfördes på ett optimalt sätt. I praktiken innebär detta att även om modeller tränade på MNIST kan vara effektiva vid att identifiera och klassificera siffror under optimala förhållanden, så kan deras prestanda variera betydligt när de utmanas med bilder som skiljer sig från de typiska format och kvalitet de är tränade på. Detta innebär att det är viktigt att noggrant överväga dessa faktorer när man förbereder bilder för prediktion för att maximera modellens noggrannhet och tillförlighet.

Sammanfattningsvis kan en utökning av dessa områden ge en heltäckande bild av de utmaningar och övervägande som är involverade i att utveckla och tillämpa maskininlärnings-modeller effektivt. Genom att ta upp dessa aspekter kan perspektivet skifta från att bara uppnå hög noggrannhet till att förstå nyanserna av att tillämpa maskininlärnings-modeller i praktiska, verkliga situationer.

# 6. Teoretiska frågor

**1. Kalle delar upp sin data i ”Träning”, ”Validering” och ”Test”, vad används respektive del för?**
Den här processen kallas för validation set metodiken. I den här metodiken är syftet att man utvärderar modellen på tre delar; Träning, Validering och Test. På **tränings delen** används datan för att träna maskininlärning modellen. Att använda tränings delen gör att modellen lär sig och anpassar sina parametrar för att göra korrekta förutsägelser eller utföra en specifik uppgift. Denna delen är den största delen av datan och är kritisk för att bygga en effektiv modell.

På **validerings delen** är syftet främst att finjustera och utvärdera modellens prestanda under träningen. Denna delen är kritisk eftersom det hjälper oss att avgöra när en modell börjar överanpassa sig till tränings setet. Överanpassning är ett fenomen där modellen presterar så pass bra på träningsdatan men dåligt på ny och osedd data. Den här delen hjälper även när det kommer till att justera modellens hyperparametrar för att förbättra sin prestanda. Hyperparametrar är inställningar på modellen som inte ändras under träning.

Den sista nämnda delen är **test delen** som är en separat del av datan som inte används under träningen av modellen. I den här delen används testdatan för att bedöma hur bra modellen presterar på helt ny data. Den används helt enkelt för att ge en slutlig prestandamätning.

**2. Julia delar upp sin data i träning och test. På träningsdatan så tränar hon tre modeller; ”Linjär Regression”, ”Lasso Regression" och en ”Random Forest modell”. Hur skall hon välja vilken av de tre modellerna hon skall fortsätta använda när hon inte skapat ett explicit ”valideringsdataset”?**
Julia kan använda cross validation. Det är en metod som tillåter henne att utvärdera modellens prestanda på olika delar av tränigns-datan för att simulera användningen av ett separat valideringsdataset. Cross validation delas upp i k-folds (vanligtvis 5 eller 10). För varje modell, tränar hon k-1 av dessa undergrupper och validerar modellen på den resterande undergruppen. Detta upprepas k gånger med en annan undergrupp som validering set varje gång. Sedan bedöms modellens prestanda som medelvärdet av dess prestanda på de olika validering uppsättningarna.

**3. Vad är "regressionsproblem"? Kan du ge några exempel på modeller som används och potentiella tillämpningsområden?**
I maskininlärning involverar regressionsproblem för att föutsäga en kontinuerlig variabel baserat på indatafunktioner. Modeller som linjär regression, polynomregression och neurala nätverk används ofta. De används för att förutsäga utfall som aktiekurser, försäljningsprognoser eller patient hälso statisktik.

**4. Hur kan du tolka RMSE och vad används det till?**

RMSE står för Root Mean Squared Error och används som mått för att mäta skillnaden mellan värden som förutses av en modell och de faktiska värdena. RMSE används som mätningsfel i bland annat modellutvärdering och parameter justering.

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

**Tolkning:**

*yi* = den faktiska värdet av den i:te observationen

*ŷi* = det förutsagda värdet för den i:te observationen

**n** = totala antal observationer

∑(*yi* − *ŷi* )**^2** = beräknar den kvadrerade skillnaden mellan de förutsagda och faktiska värdena.

√∑(*yi* − *ŷi* )**^2** = ger ett mått på samma enheter som den beroende variabeln, gör det lättare att tolka än ett mått som kvadrerade fel.

**5. Vad är ”klassificieringsproblem? Kan du ge några exempel på modeller som används och potentiella tillämpningsområden? Vad är en ”Confusion Matrix”?**

Ett klassificeringsproblem kategoriserar data i fördefinierade klasser. modeller som Logistisk Tegression och Neurala Nätverk används ofta. Tillämpningar inkluderar spam filter i mail och bildigenkänning. En confusion matrix visar noggranheten hos en modell genom att visa sanna positiva, falska positiva, sanna negativa och falska negativa, vilket hjälper till att utvärdera prestanda.

**6. Vad är K-means modellen för något? Ge ett exempel på vad det kan tillämpas på.**

K-means är en klusterings-modell som är en typ av oövervakad inlärning. Detta innebär att den används när man inte har märkta svar i datan. K-means syftar till att grupper liknande datapunkter tillsammans i kluster istället för att förutsäga en variabel. Modellen kan tillämpas i bland annat kundsegmentering och bildsegmentering

**7. Förklara (gärna med ett exempel): Ordinal encoding, one-hot encoding, dummy variable encoding.**

Ordinal encoding omvandlar kategoridata till heltal koder baserat på ordning t.ex. bety ”låg”, ”medel”, ”hög” till 1,2,3. One hot encoding omvandlar kategoridata till binära vektorer som representerar varje kategori: för ”röd”, ”blå”, ”grön” skapar den tre separata kolumner. Dummy variable encoding liknar one hot men sjunker en nivå för att undvika multicollinearity, så ”röd”, ”blå”, ”grön”, kan den släppa grön och använda två kolumner istället. Dessa metoder hjälper till att förbereda kategoriska data för maskininlärnings-modeller.

**8. Göran påstår att datan antingen är ”ordinal” eller ”nominal”. Julia säger att detta måste tolkas. Hon ger ett exempel med att färger såsom {röd, grön, blå} generellt sett inte har någon inbördes ordning (nominal) men om du har en röd skjorta så är du vackrast på festen (ordinal) – vem har rätt?**
Göran och Julia belyser två olika kategoriska data. Göran påpekar att data kan vara ordinal eller nominal. Julia hävdar att klassificering beror på sammanhanget: medan färgerna i sig kan vara nominella blir de ordinära i vissa scenarier, som hennes exempel. Båda är korrekta; typen av data bestäms av dess användning och tolkning i specifika sammanhang.

**9. Vad är Streamlit för något och vad kan det användas till?**
Streamlit är en open-source designat för att utveckla interaktiva webbapplikationer i Python programmering miljö. Det gör det möjligt att omvandla dataanalys skript till användarvänliga appar med minimal webbutveckling upplevelse. Används flitigt inom maskininlärning och datavetenskap. Streamlit underlättar skapandet av applikationer för datavisualisering, utforskning och interaktiv modelldemonstration.

# Källförteckning

IBM, (ingen datum), What is logistic regression?, Tillgänglig på: https://www.ibm.com/topics/logistic-regression (hämtad 22/03-24)

IBM, (ingen datum), What is random forest?, Tillgänglig på: (https://www.ibm.com/topics/logistic-regression (hämtad 22/03-24)

IBM, (ingen datum), What are support vector machines (SVMs)?, Tillgänglig på: https://www.ibm.com/topics/support-vector-machine?mhsrc=ibmsearch_a&mhq=support%20vector%20machine (hämtad 22/03-24)

Great Learning, (publicerad 30/5-23), Hyperparameter Tuning with GridSearchCV, Tillgänglig på: https://www.mygreatlearning.com/blog/gridsearchcv/ (hämtad 22/03-24)