Chronic Kidney Disease Prediction

Machine Learning Classification Assignment

Fawad Saleem

Dataset	CKD.csv — Chronic Kidney Disease
Records	399 Patients | 25 Features
Best Model	Support Vector Machine (SVM)
Best F1 Score	1.0 (100% — Perfect Classification)
Models Compared	Decision Tree | Random Forest | SVM | Logistic Regression

1 Problem Statement

The aim is to predict Chronic Kidney Disease (CKD) based on several clinical and laboratory parameters collected from patients. The model classifies each patient as either **CKD positive** or **CKD negative** based on their health measurements.

2 Dataset Information

Property	Details
File	CKD.csv
Total Rows	399 (one row per patient)
Total Columns	25 (health parameters)
Target Column	classification (CKD / Not CKD)
CKD Positive	249 patients (True)
CKD Negative	150 patients (False)

3 Pre-processing Method

The dataset contains several categorical (text) columns with values like **yes/no** or **normal/abnormal**. Since machine learning models only work with numbers, these were converted using **One-Hot Encoding** via `pd.get_dummies(drop_first=True)`.

Columns converted (Nominal Data — Yes/No):

• rbc • pc • pcc • ba • htn • dm • cad • appet • pe • ane

Step	Action	Tool Used
1	Load CSV data	pd.read_csv()
2	One-Hot Encode categorical columns	pd.get_dummies(drop_first=True)
3	Split features (X) and target (y)	Manual column selection
4	Train/Test split (67% / 33%)	train_test_split(test_size=1/3)
5	Feature scaling	StandardScaler

4 Final Model — Support Vector Machine (SVM)

SVM was selected as the final model because it achieved the highest F1-macro score of **1.0** compared to all other models tested, indicating perfect classification on the test set. GridSearchCV was used to find the optimal hyperparameters.

Hyperparameter	Best Value	Meaning

kernel	poly	Polynomial kernel — captures non-linear patterns
C	10	Regularization — allows some misclassification flexibility
gamma	auto	Kernel coefficient — auto-set based on features
scoring	f1_weighted	Optimized for weighted F1 score
cv	5 folds	Cross-validation folds used in GridSearchCV

Confusion Matrix — SVM:

```
[[51  0]
 [ 0 82]]
              precision    recall  f1-score   support

       False       1.00      1.00      1.00        51
        True       1.00      1.00      1.00        82

    accuracy                           1.00       133
   macro avg       1.00      1.00      1.00       133
weighted avg       1.00      1.00      1.00       133
```

Figure 1: SVM Confusion Matrix — Perfect classification (0 errors)

F1-Macro Score	**1.0 (100%)**	**Accuracy**	**100%**

5 Model Comparison

Four models were trained and evaluated using GridSearchCV with 5-fold cross-validation. All models performed well, but SVM achieved a perfect score.

Model	F1 Score	Best Hyperparameters
Decision Tree	0.977	`criterion='entropy', max_features='log2', splitter='random'`
Random Forest	0.985	`criterion='entropy', max_features='log2', n_estimators=100`
Logistic Regression	0.992	`penalty='l2', solver='newton-cg'`
SVM ★ BEST	**1.000**	`C=10, gamma='auto', kernel='poly'`

Confusion Matrices — All Models:

```
[[50  1]
 [ 2 80]]
              precision    recall  f1-score   support

       False       0.96      0.98      0.97        51
        True       0.99      0.98      0.98        82

    accuracy                           0.98       133
   macro avg       0.97      0.98      0.98       133
weighted avg       0.98      0.98      0.98       133
```

Figure 2: Decision Tree (F1=0.977)

```
[[50  1]
 [ 1 81]]
              precision    recall  f1-score   support

       False       0.98      0.98      0.98        51
        True       0.99      0.99      0.99        82

    accuracy                           0.98       133
   macro avg       0.98      0.98      0.98       133
weighted avg       0.98      0.98      0.98       133
```

Figure 3: Random Forest (F1=0.985)

```
[[51  0]
 [ 1 81]]
              precision    recall  f1-score   support

       False       0.98      1.00      0.99        51
        True       1.00      0.99      0.99        82

    accuracy                           0.99       133
   macro avg       0.99      0.99      0.99       133
weighted avg       0.99      0.99      0.99       133
```

Figure 4: Logistic Regression (F1=0.992)

```
[[51  0]
 [ 0 82]]
              precision    recall  f1-score   support

       False       1.00      1.00      1.00        51
        True       1.00      1.00      1.00        82

    accuracy                           1.00       133
   macro avg       1.00      1.00      1.00       133
weighted avg       1.00      1.00      1.00       133
```

Figure 5: SVM — Best Model (F1=1.000)

F1 Score Comparison:

Decision Tree	**0.977**		
Random Forest	**0.985**		
Logistic Regression	**0.992**		
SVM (Best)	**1.000**		★

6 Model Deployment

The trained SVM model was saved using `pickle` and deployed in a separate Jupyter Notebook. Two prediction samples are demonstrated below.

```
import pickle

# Save model and scaler
pickle.dump(grid, open("finalised_model_SVM.sav", "wb"))
pickle.dump(sc, open("scaler_SVM.sav", "wb"))

# Load in deployment notebook
loaded_model = pickle.load(open("finalised_model_SVM.sav", "rb"))
loaded_scaler = pickle.load(open("scaler_SVM.sav", "rb"))
```

Sample 1 — Hardcoded Input (CKD Patient)

Input Values:

```
patient = [[48, 80, 4, 0, 200, 100, 7.2, 130, 5.5, 8.5, 28, 11000, 3.2,
            0,  1,  0, 0,  0,  0,   1,  0,   1,  1,  0,    0,  1,  1]]
```

Figure 6: Hardcoded patient array — CKD positive patient

Output:

```
[ True]
CKD DETECTED
```

Figure 7: Prediction result — CKD Detected

Sample 2 — User Input via Prompt (CKD Patient)

Input Values (entered by user):

```
Age:  48
Blood Pressure:  80
Albumin (0-5):  4
Sugar (0-5):  0
Blood Glucose:  200
Blood Urea:  100
Serum Creatinine:  7.2
Sodium:  130
Potassium:  5.5
Hemoglobin:  8.5
Packed Cell Volume:  28
White Blood Cell Count:  11000
Red Blood Cell Count:  3.2

 Enter 1 for Yes, 0 for No

Specific Gravity 1.010:  0
Specific Gravity 1.015:  1
Specific Gravity 1.020:  0
Specific Gravity 1.025:  0
Red Blood Cells Normal:  0
Pus Cells Normal:  0
Pus Cell Clumps Present:  1
Bacteria Present:  0
Hypertension:  1
Diabetes:  1
Coronary Artery Disease:  0
Good Appetite:  0
Pedal Edema:  1
Anemia:  1
```

Figure 8: User-entered patient details

```
--- RESULT ---
CKD DETECTED
CKD Chance     : 100.00%
No CKD Chance : 0.00%
```

Figure 9: Prediction Result

7 Conclusion

• **Four models were compared:** Decision Tree, Random Forest, Logistic Regression, and SVM — all using GridSearchCV hyperparameter tuning.

• **SVM was selected as the best model:** It achieved a perfect F1-macro score of 1.0, outperforming all other models.

• **Pre-processing was applied:** One-Hot Encoding converted categorical columns to numerical format; StandardScaler normalized all features.

• **Successful deployment:** The model was saved using pickle and successfully deployed in a separate notebook with both hardcoded and user-input prediction modes.

• **Note on perfect score:** The 1.0 score reflects the clean and relatively small nature of this dataset (399 rows). Real-world performance may vary with noisier, larger clinical data.