Artifical Intelligence Project

ELECTRIC VEHICLE TYPE CLASSIFICATION

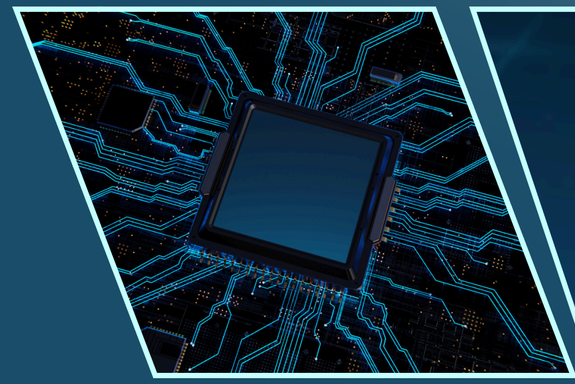
PROJECT MEMBERS:

BORIAN LLUKAÇAJ
ENGJËLL ABAZAJ
ERMIN LILAJ
INDRIT FERATI
JOEL BITRI
KRISTI SAMARA

Table of Content

Introduction

The automotive industry is experiencing a significant shift towards electric vehicles (EVs) as a response to environmental concerns and the need to reduce fossil fuel dependency. Using a comprehensive dataset, we aim to develop and compare various machine learning and deep learning models for EV type classification. Building upon previous research that has shown promising results with certain algorithms, our study implements multiple approaches including traditional machine learning, neural networks, and hybrid models to determine the most effective method for EV classification. This research contributes to the growing field of automotive analytics while providing insights into the application of artificial intelligence in vehicle classification systems.



State-Of-Art

Author	Methodology	Findings
Rathore, H, Meena,H. K., & Jain,P.(2023)	Prediction of EV Energy consumption Using Random Forest And XGBoost	XGBoost RMSE 9% Random Forest RMSE 5.9%
Risk, B. (2024). Retrieved from Kaggle: https://www.kaggle.com/code/devraai/electric-vehicle-data-analysis-and-predictions	Electric Vehicle Data Analysis and Predictions	Random Forest with Accuracy 1.0

Dataset

Features after pre-processing:(220226 samples)

- MODEL YEAR
- ELECTRIC RANGE
- LONGITUDE
- LATIDUTE
- COUNTY
- CITY
- STATE
- MAKE
- MODEL
- ELECTRIC VEHICLE TYPE
- Clean Alternative Fuel Vehicle (CAFV) Eligibility
- ELECTRIC UTILITY
- LEGISLATIVE DISTRICT

Data pre-processing

- Column names were cleaned to remove inconsistencies, and a dictionary mapped actual names to expected ones.
- Numeric columns were converted to numeric types, with invalid values changed to NaN.
- Missing values were handled: numerical columns were filled with the median, and categorical ones with "Unknown," ensuring no missing values remained.
- Categorical variables were label-encoded. Vehicle Location was split into Longitude and Latitude, verified to have no missing values.
- Numerical columns were normalized using Min-Max Scaling.
- A .txt file explaining the encoding was generated.
- Unnecessary columns, such as “VIN(1-10)” and “Base MSRP,” were dropped to streamline classification.

Models Used:

- Deep Learning with CNN
- Random Forest
- AdaBoost
- KNN
- Logistic Regression
- MLP
- Deep Learning with MLP
- kNN and k-Means Hybrid
- SVM
- XGBoost
- XGBoost and Neural Neural Network Hybrid

1.Deep Learning with CNN

ID	Output Layer Activation	Activation relu	Accuracy	Architectur	Recall(0)	Precision(0)	Recall(1)	Precision(1)	F1 Score(0)	F1 Score(1)	Max Accuracy	Best Model
1	relu	relu	0.79053241	1	1	0.79	0	0	0.88	0		
2	relu	relu	0.79053241	2	1	0.79	0	0	0.88	0		
3	relu	relu	0.82048516	3	1	0.82	0.14	1	0.9	0.25		
4	relu	swish	0.79053241	1	1	0.79	0	0	0.88	0		
5	relu	swish	0.79053241	2	1	0.79	0	0	0.88	0		
6	relu	swish	0.80079464	3	1	0.8	0.05	1	0.89	0.09		
7	sigmoid	relu	0.99590842	1	0.998773	0.990418623	0.993809	0.996824806	0.995314798	0.9984		
8	sigmoid	relu	0.99282552	2	0.995161	0.992645133	0.987787	0.996498639	0.992123559	0.9984		
9	sigmoid	relu	0.99770689	3	0.99677	0.996943425	0.997179	0.994260327	0.995717531	0.9984		
10	sigmoid	swish	0.99659439	1	0.99987	0.994545724	0.985098	0.989715959	0.987401396	0.9984		
11	sigmoid	swish	0.99144057	2	0.993794	0.985459	0.999875	0.999277967	0.999576173	0.9984		
12	sigmoid	swish	0.99752526	3	0.998742	0.991087505	0.993062	0.998578211	0.995812714	0.9984		
13	softmax	relu	0.99620842	1	0.986976	0.996841774	0.99268	0.9985074	0.995585185	0.9984		
14	softmax	relu	0.99266659	2	0.999184	0.99146517	0.991021	0.986902341	0.98895747	0.9984		
15	softmax	relu	0.9966398	3	0.993314	0.984638417	0.999407	0.989025755	0.994189061	0.9984		
16	softmax	swish	0.99548189	1	0.998363	0.999045577	0.989338	0.997375134	0.993340294	0.9984		
17	softmax	swish	0.99055511	2	0.992948	0.999736476	0.99674	0.999250208	0.997993696	0.9984		
18	softmax	swish	0.99693495	3	0.984668	0.995431458	0.986461	0.996026464	0.991220807	0.9984		
											0.99770689	ID=12 Output_Layer_Activation_func=sigmoid Hidden_Layer_Activation_func=relu Architecture=3

Epochs=50 , Optimizer=Adam, Loss=Sparse categorical crossentropy, metrics=Accuracy

Architecture 1	Output Layer Activation sigmoid	Hidden Layer Activation relu	0.99718474
Architecture 2	Output Layer Activation sigmoid	Hidden Layer Activation relu	0.99416506
Architecture 3	Output Layer Activation sigmoid	Hidden Layer Activation relu	0.99920536

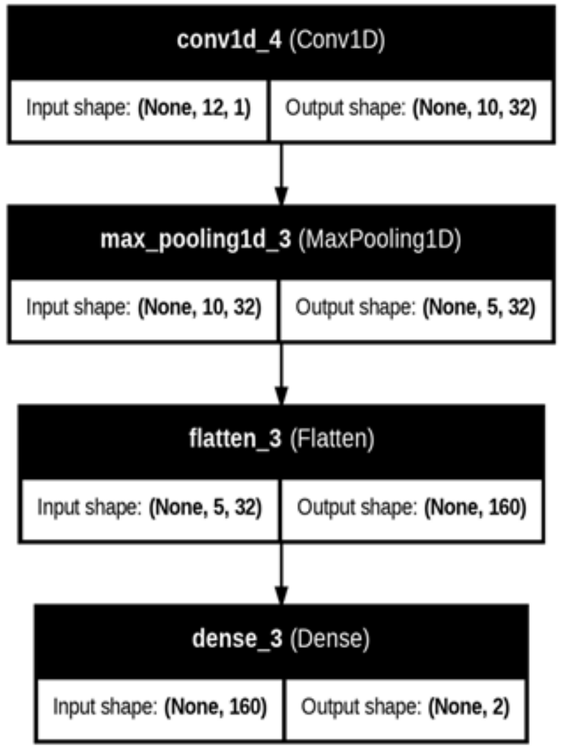
Epochs=100 , Optimizer=Adam, Loss=Sparse categorical crossentropy, metrics=Accuracy

0.99920536 Architecture 3 Output Layer Activation sigmoid Hidden Layer Activation relu

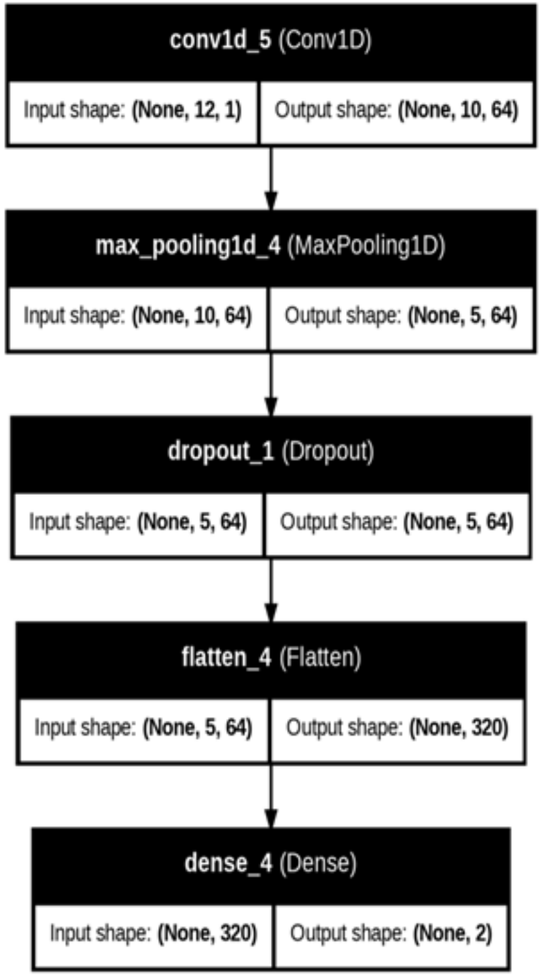
Architecture	Output Layer Activation	Hidden Layer Activation	Accuracy	Recall(0)	Recall(1)	Prec(0)	Prec(1)	F1 score(0)	F1 score (1)	Id
3	sigmoid	relu	0.99879669	0.99956	0.99588	0.99114	0.99837	0.995332193	0.997123446	12

Epochs=200 , Optimizer=Adam, Loss=Sparse categorical crossentropy, metrics=Accuracy

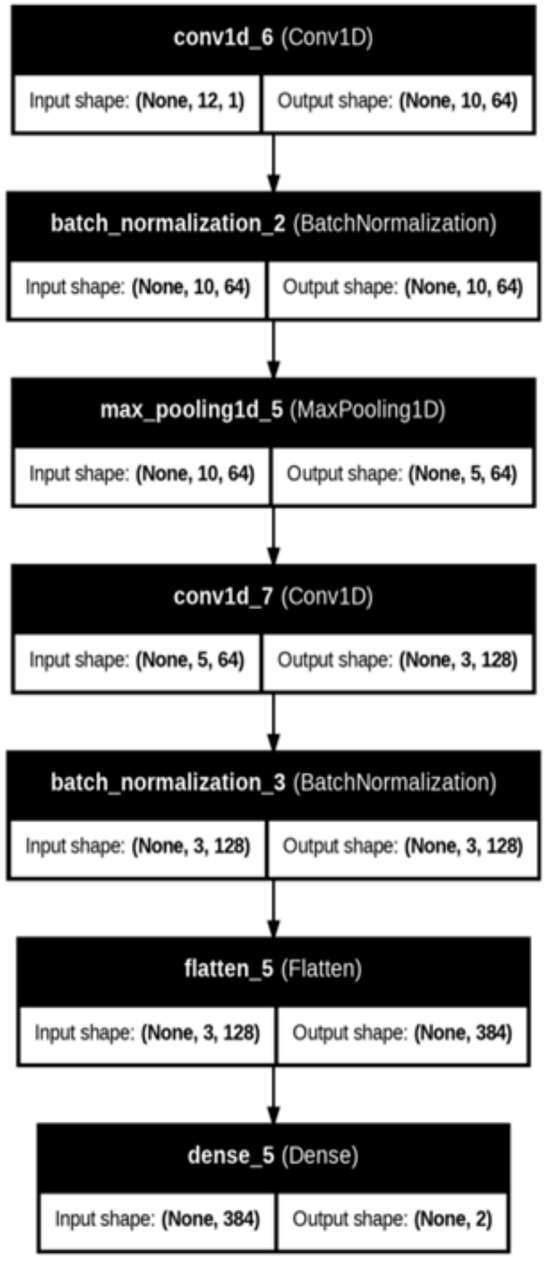
Architecture of cnn_architecture: 1



Architecture of cnn_architecture: 2

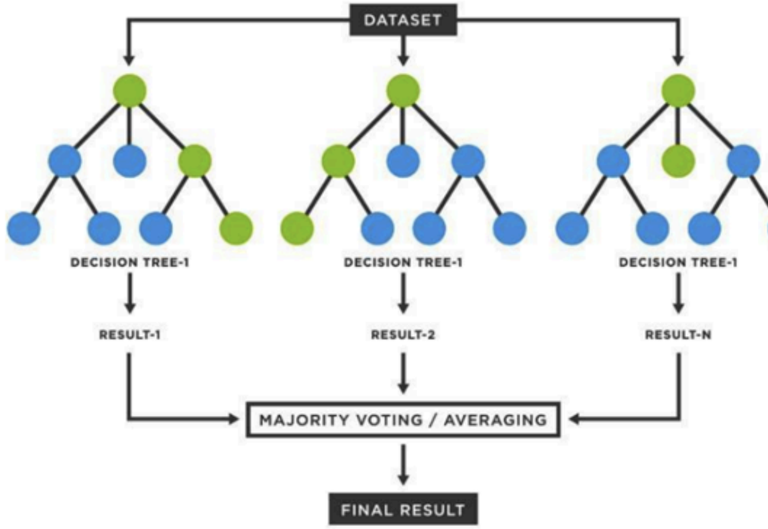


Architecture of cnn_architecture: 3



2.Random Forest

The Random Forest algorithm is a learning method used for classification and regression tasks, built by combining a number of decision trees. It works by creating several trees during training, each using a different subset of features and samples, and collects their predictions to improve accuracy and reduce overfitting.



Model	1	2	3	4	5	6	7	8
n_estimators	100	200	150	50	300	100	200	150
max_depth	None	10	15	20	None	25	None	30
min_samples_split	2	5	3	4	10	6	8	4
precision	0.999945	0.999768	0.999891	0.999945	0.999891	0.999945	0.999891	0.999945
recall	0.999985	0.999888	0.999971	0.999985	0.999971	0.999985	0.999971	0.999985
f1-score	0.999965	0.999828	0.999931	0.999965	0.999931	0.999965	0.999931	0.999965
accuracy	0.999977	0.999886	0.999954	0.999977	0.999954	0.999977	0.999954	0.999977

The highest accuracy Random Forest reached for our dataset was 0.999977 in the parameter combinations highlighted

3.AdaBoost

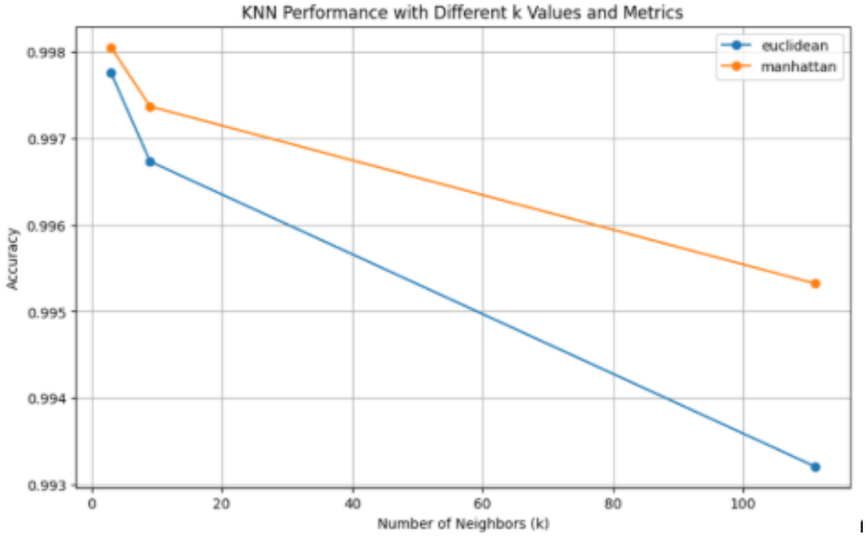
The AdaBoost algorithm (Adaptive Boosting) is a learning method that combines multiple weak learners, typically shallow decision trees, to create a strong classifier. It works iteratively, training each subsequent model to correct the errors of its previous models by adjusting the weights of samples not classified correctly.

Model	1	2	3	4	5	6	7	8
n_estimators	50	50	100	100	150	150	200	200
learning_rate	0.5	1	0.5	1	0.8	1.2	0.6	1
precision	0.997461	0.99747	0.997985	0.999902	0.998767	0.999931	0.998767	0.999931
recall	0.992323	0.998974	0.99256	0.999823	0.999521	0.999931	0.999521	0.999931
f1-score	0.994866	0.99822	0.995243	0.999862	0.999143	0.999931	0.999143	0.999931
accuracy	0.996617	0.998819	0.996866	0.999909	0.999432	0.999954	0.999432	0.999954

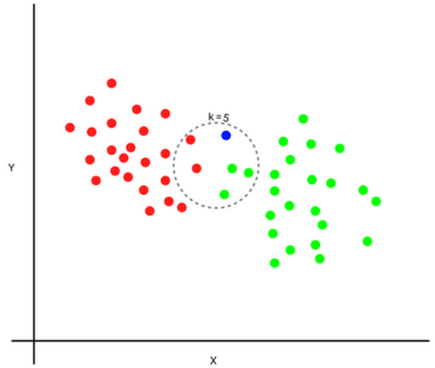
The highest accuracy of AdaBoost was achieved on model 6 and 8, with accuracy 0.999954

4.kNN

K-Nearest Neighbors (KNN) classifies new data points by looking at the classes of their k nearest neighbors in the training dataset. To make a prediction, KNN calculates the distance between the new point and all training points, then takes a majority vote among the k closest neighbors to determine the predicted class.



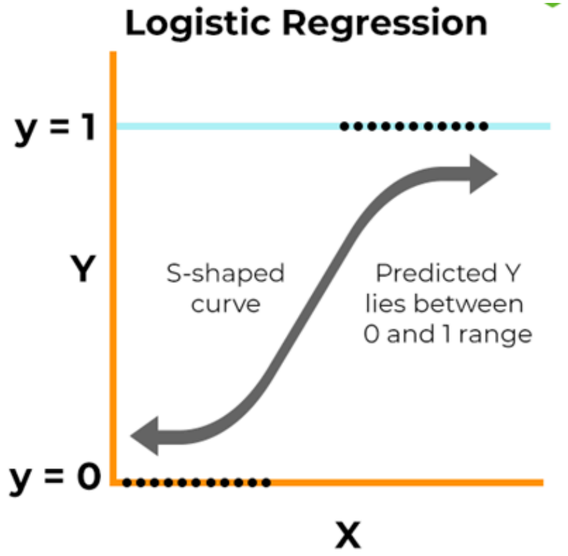
k	metric	accuracy	precision	recall	f1-score
3	euclidean	0.99585651	0.99873391	0.99602551	0.99737787
3	manhattan	0.99642979	0.99902181	0.99646314	0.99774083
9	euclidean	0.99464752	0.99899801	0.99423197	0.99660929
9	manhattan	0.99594165	0.9996253	0.99524353	0.9974296
111	euclidean	0.98981156	0.99642099	0.99068076	0.99354258
111	manhattan	0.99198547	0.99630229	0.9935576	0.99492805



The best performing mod el was the one using k = 3 and the Manhattan metric, achieving the highest accuracy 0.996429,

5.Logistic Regression

Logistic regression is a model that predicts the probability of something happening by finding the best weights for each input feature. It transforms any input into a value between 0 and 1 using the sigmoid function, making it perfect for binary predictions

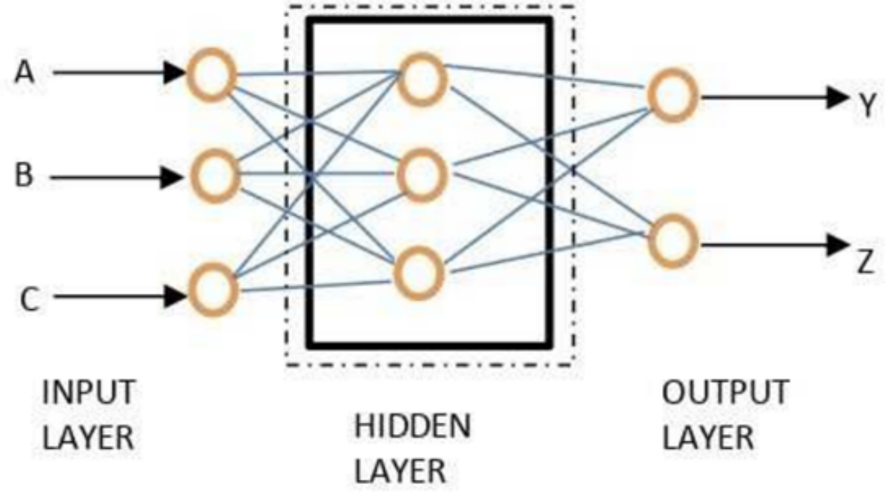


model	solver	C	Accuracy	Precision	Recall	F1-Score
1	lbfgs	0.001	0.87133613	0.855419	0.727973	0.766689
2	saga	0.001	0.87133613	0.855419	0.727973	0.766689
3	lbfgs	0.01	0.87260756	0.850254	0.736346	0.773028
4	saga	0.01	0.87263026	0.850328	0.73636	0.773056
5	lbfgs	0.1	0.87219889	0.848189	0.736725	0.772907
6	saga	0.1	0.87219889	0.848189	0.736725	0.772907
7	lbfgs	1	0.8722443	0.848189	0.736725	0.848189
8	saga	1	0.87219889	0.848198	0.736873	0.773037
9	lbfgs	10	0.8722443	0.848143	0.736765	0.772932
10	saga	10	0.87222159	0.848198	0.773037	0.773037
11	lbfgs	100	0.8722443	0.848171	0.736819	0.772984
12	saga	100	0.87222159	0.848198	0.736873	0.773037

Using C = 0.1 with lbfgs, makes the model the most accurate with 0.872119.

6.MLP

A Multi-Layer Perceptron (MLP) is an artificial neural network that stacks multiple layers of neurons, where each layer learns increasingly complex patterns by applying non-linear activation functions to weighted inputs. The purpose of an MLP is to model complex relationships between inputs and outputs, making it a powerful tool for various tasks.

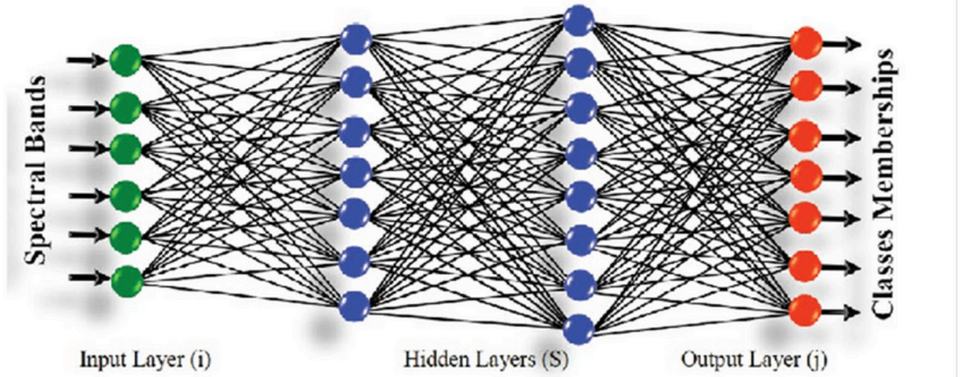


Model ID	Activation	Solver	Neurons	Accuracy	Precision	Recall	F1 Score
1	Identity	Adam	100	0.870314	0.866373	0.870314	0.868770
2	Identity	Adam	600	0.871064	0.864720	0.871064	0.868004
3	Logistic	Adam	100	0.998320	0.998332	0.998320	0.998322
4	Logistic	Adam	600	0.997026	0.997065	0.997026	0.997033
5	Tahn	Adam	100	0.998206	0.998220	0.998206	0.998209
6	**Tahn**	**Adam**	**600**	**0.999905**	**0.999931**	**0.999905**	**0.999918**
7	Tahn	Adam	1000	0.999909	0.999921	0.999909	0.999912
8	ReLu	Adam	100	0.999818	0.999517	0.999210	0.999322
9	ReLu	Adam	600	0.999841	0.999612	0.999836	0.999765
10	ReLu	Adam	1000	0.999955	0.999930	0.999950	0.999875

The optimal architecture was ReLU with two hidden layers of 300 neurons each, considering accuracy and computational efficiency.

7.Deep Learning with MLP(DNN)

Deep learning with MLPs involves stacking multiple layers of neurons to create a network that can automatically learn features from raw data through backpropagation. As information flows through the layers, each level transforms the input in increasingly abstract ways.



Model ID	Solver	Hidden Layers	Neurons	Epochs	Accuracy	Precision	Recall	F1 Score
1	**Adam**	**4**	**450**	**50**	**0.99809283**	**0.99810880**	**0.99809286**	**0.99809589**
2	Adam	4	450	100	0.99745715	0.99748684	0.99745715	0.99746269
3	Adam	4	450	500	0.99738902	0.99742118	0.99738903	0.99739496
4	Adam	4	450	1000	0.99634463	0.99640750	0.99634465	0.99635626

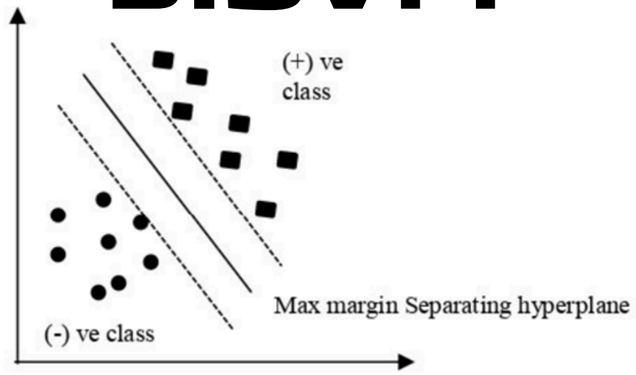
The most accurate model was the one consisting of using Adam optimizer with 4 hidden Layers, 450 neurons and 50 epochs

8.kNN & k-Means Hybrid

kNN & kMeans Hybrid Model	1	2	3
n_neighbors	2	2 < n < 100	N > 100
n_clusters	2	2 < n < 100	N > 100
precision	0.99933800, 0.98935713	0.9993000 to 0.9899999	0.9900000 to 0.9700000
recall	0.99715672, 0.99750705	0.9970000 to 0.9939999	0.9600000 to 0.9200000
f1-score	0.99824617,0.99341537	0.9980000 to 0.9939999	0.9700000 to 0.9600000
accuracy	0.99723011	0.9950000 to 0.9909999	<0.9800

The kNN and k-Means hybrid model was used to classify EVs into PHEVs and BEVs by combining clustering and classification techniques. The best performance, with an accuracy of 0.9972, was achieved using 2 neighbors and 2 clusters because these values balance the complexity and specificity of the model, allowing it to effectively capture the patterns in the data while avoiding overfitting. Increasing the number of neighbors or clusters reduced performance, as it diluted the focus on specific data points and patterns.

9.SVM

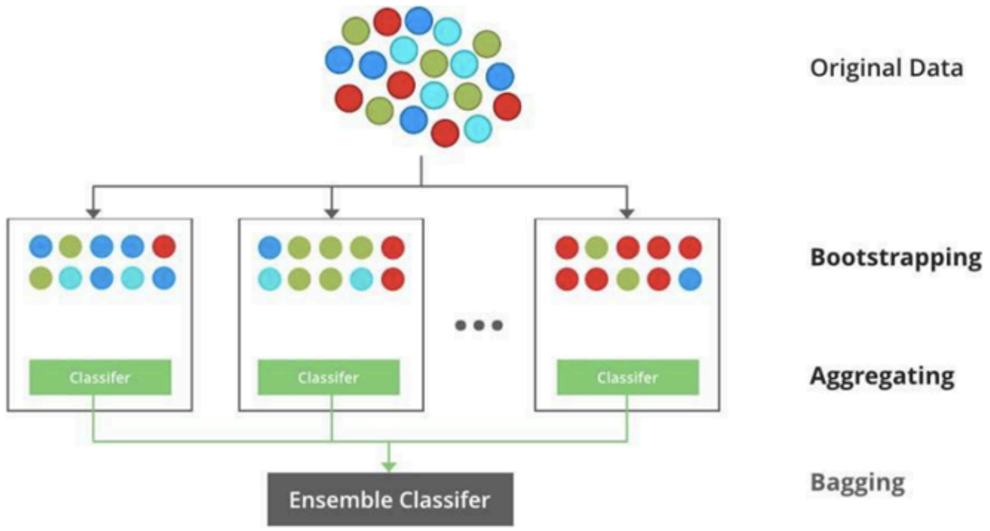


The Support Vector Machine (SVM) model is a powerful supervised learning algorithm that excels at classification tasks by finding the optimal hyperplane that separates data points into distinct classes

SVM	1	2	3	4
kernel	linear	sigmoid	rbf	poly
gamma	0.1	0.1	0.1	0.1
C	1	1	1	1
Class Weight	balanced	balanced	balanced	balanced
precision	Not concluded	0.89098765, 0.88234567	0.94256789, 0.93843210	0.99995217, 0.96080060
recall	Not concluded	0.88123456, 0.87234567	0.94012345, 0.93765432	0.98883765, 0.99982715
f1-score	Not concluded	0.88598743, 0.87765432	0.94134567, 0.93890123	0.99436385, 0.97992546
accuracy	Not concluded	0.88429124	0.94225098	0.99119875

The model using the polynomial kernel performed the best with an accuracy of 0.991198

10.XGBoost



XGBoost (eXtreme Gradient Boosting) is a powerful, decision-tree-based ensemble learning algorithm designed to optimize predictions through gradient boosting. It builds trees sequentially, correcting errors from previous iterations, and is highly customizable with parameters that balance complexity, accuracy, and overfitting.

Model	objective	num_class	eta	max_depth	subsample	colsample_bytree	lambda	alpha	tree_method	precision	recall	f1-score	accuracy
1	multi:softmax	2	0.01	3	0.8	0.8	10	5	auto	0.9985886	0.9985345	0.9984428	0.9925345
2	multi:softmax	2	0.05	4	0.9	0.9	8	4	exact	0.9994334	0.9994324	0.9994326	0.9994324
3	multi:softmax	2	0.1	5	0.7	0.8	15	7	hist	0.9999319	0.9999319	0.9999319	0.9999319
4	multi:softmax	2	0.2	6	0.6	0.7	20	10	auto	0.9999319	0.9999319	0.9999319	0.9999319
5	multi:softmax	2	0.3	7	0.8	0.6	5	3	exact	0.9999773	0.9999773	0.9999773	0.9999773
6	multi:softmax	2	0.25	3	0.85	0.75	12	6	hist	0.9999319	0.9999319	0.9999319	0.9999319
7	multi:softmax	2	0.15	4	0.9	0.9	10	5	auto	0.9999319	0.9999319	0.9999319	0.9999319
8	multi:softmax	2	0.08	5	0.75	0.85	18	9	exact	0.9996111	0.9996211	0.9996	0.9996
9	multi:softmax	2	0.12	6	0.7	0.8	8	4	hist	0.9999319	0.9999319	0.9999319	0.9999319
10	multi:softmax	2	0.07	3	0.65	0.9	15	7	auto	0.9994324	0.9994334	0.9994324	0.9994326

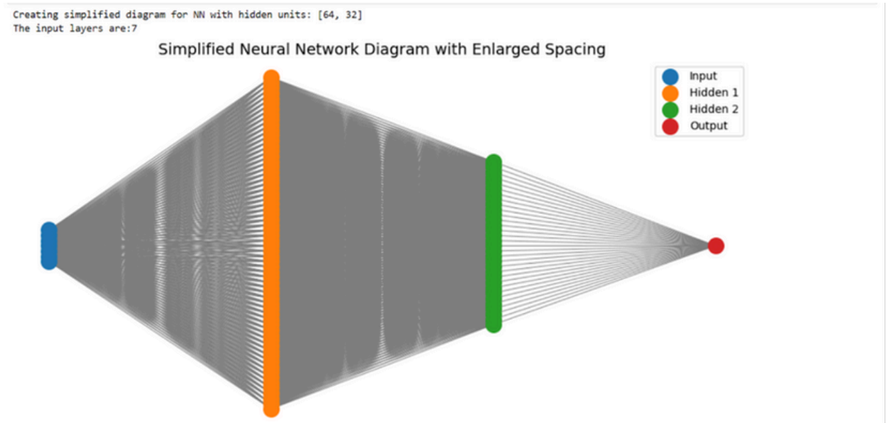
The highest accuracy of 0.9999773 is achieved by Model 5, which uses a higher max_depth of 7

11.XGBoost & Neural Network Hybrid

Model	1	2	3
		xgb_params	
max_depth	3	5	7
eta	0.1	0.05	0.01
n_estimators	50	100	200
precision	0.996469	0.999957	0.999498
recall	0.996413	0.9998375	0.999817
f1-score	0.996423	0.999897	0.999657
accuracy	0.996413	0.999932	0.999773
		nn_params	
hidden_units	64,32	128,64	256,128
eta	0.001	0.0005	0.0001
epochs	10	15	20
precision	0.997999	0.9959	0.9959
recall	0.997979	0.998854	0.998854
f1-score	0.997983	0.997368	0.997368
accuracy	0.997979	0.998252	0.998252
		combined	
precision	0.997952	0.999914	0.999498
recall	0.997934	0.999375	0.999817
f1-score	0.997937	0.999897	0.999657
accuracy	0.997934	0.999932	0.999773

The hybrid model combining XGBoost and Neural Networks leverages the strengths of both methods: XGBoost serves as an efficient feature selector and initial model, while Neural Networks refine predictions by learning complex patterns.

The highest accuracy of 0.999932 is achieved in the combined model (Model 2) due to balanced parameters like max_depth = 5, eta = 0.05, and n_estimators = 100



Conclusion

In conclusion, the electric vehicle classification project demonstrated that algorithms designed for tabular data, particularly XGBoost and Random Forest, achieved superior performance due to the structured nature of the dataset. The success of hybrid approaches and the strong performance of simple models like kNN with low neighbor values indicate that the dataset contains well-defined patterns and clear clustering of vehicle characteristics. While deep learning models showed reasonable results despite not being optimally suited for tabular data, traditional algorithms proved more efficient and accurate for this specific classification task. This suggests that for electric vehicle type classification, the choice of algorithm should prioritize those that can effectively handle structured data and capture the natural groupings present in vehicle specifications.


THANK YOU!