INTRODUZIONE AL MACHINE LEARNING

FACCHIN ALBERTO

MICHELUZ LEONARDO

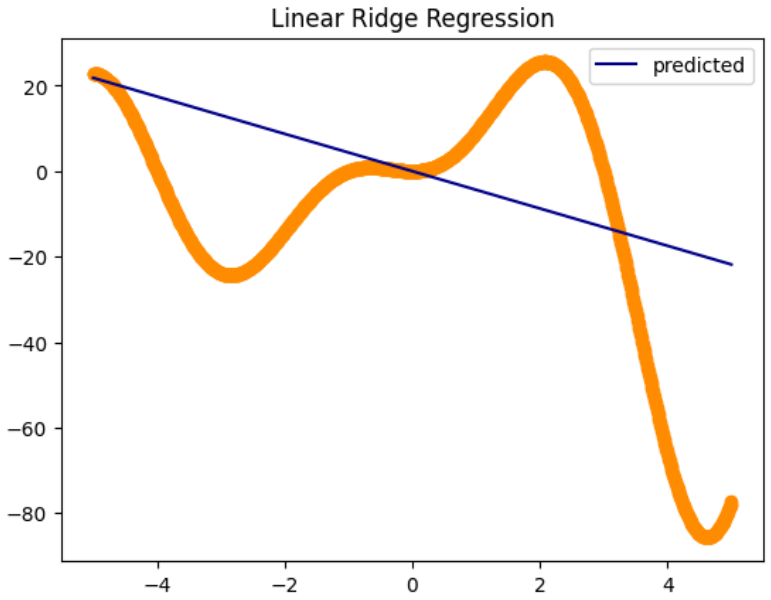
REPORT CHALLENGE DUE

Data visualization

Come sempre, il primo passo da svolgere quando si importa (o in questo caso genera) un dataset è quello di analizzarlo in maniera tale da comprendere al meglio i dati con cui si deve lavorare. Si inizia quindi istanziando 1000 osservazioni, per poi generare i due classici dataset: training set e testing set. Plottando le features e labels del training set notiamo come i punti seguano un andamento polinomiale, rendendo quindi difficile l'utilizzo di tecniche di fitting lineari.

Linear Ridge Regression

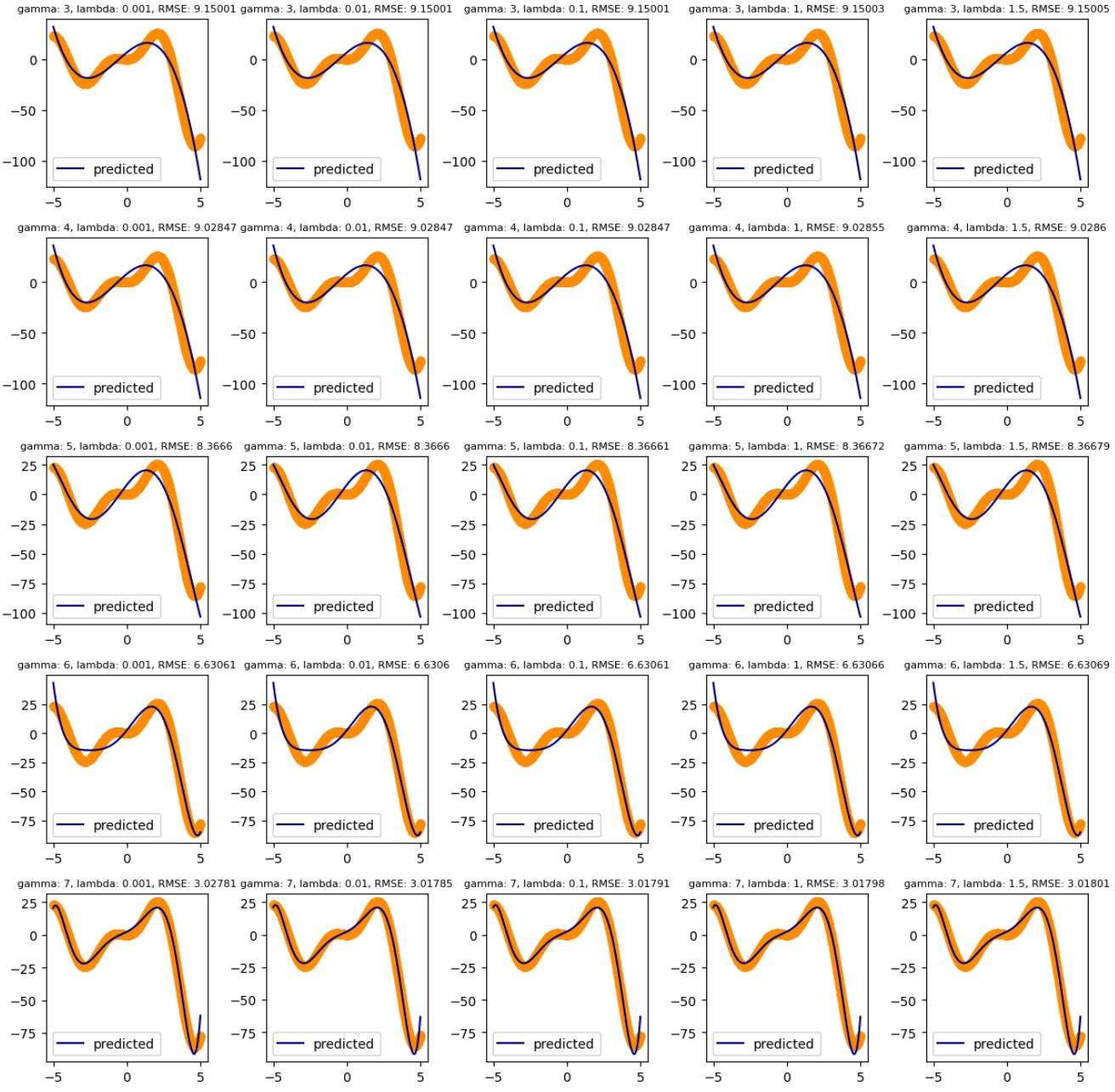
Come richiesto dalla consegna, procediamo alla creazione di un modello di Ridge Regression lineare tramite la funzione KRRS, specificando come parametro il tipo di kernel da utilizzare, in questo caso 'linear'. Successivamente calcoliamo il Root Mean Squared Error associato al nostro modello, ottenendo un valore di 28.58681. Come ci aspettavamo, il RSME è estremamente elevato in quanto abbiamo provato a fittare un dataset con complessità polinomiale (di grado maggiore ad uno) con un modello lineare. Plottando il dataset e il nostro modello lineare si può facilmente intuire come si debba passare ad un modello migliore.



Polynomial Ridge Regression

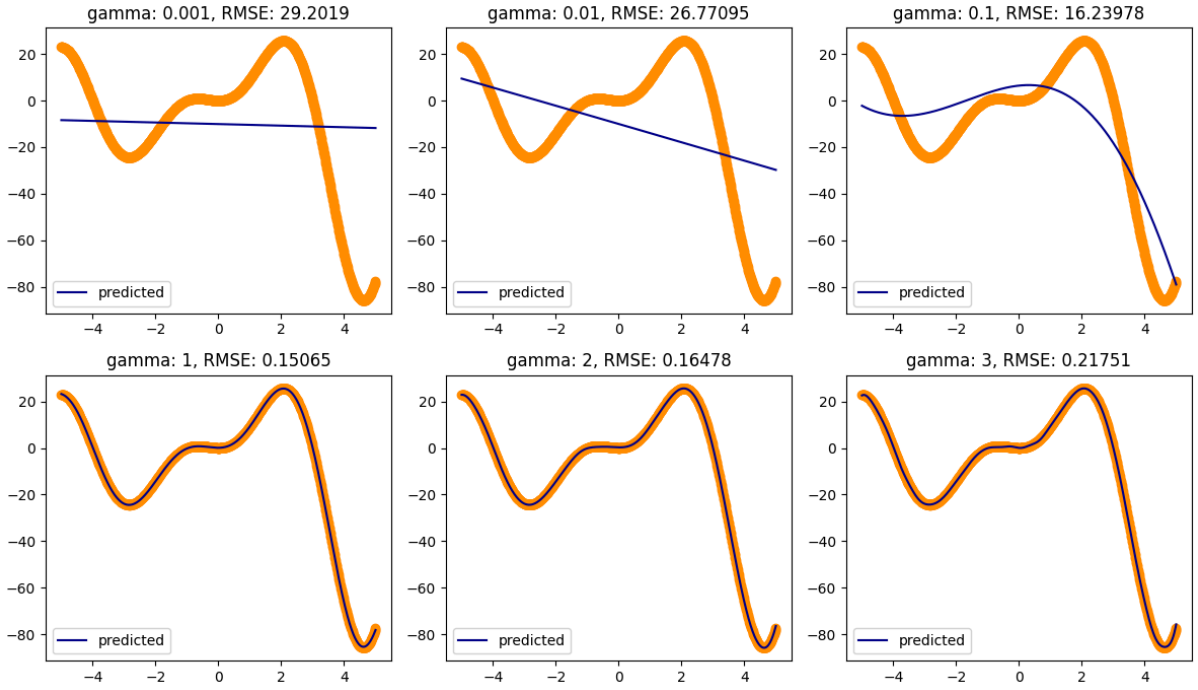
Proviamo quindi ad utilizzare un modello di Ridge Regression con kernel polinomiale, verificando quindi se si ottiene un risultato migliore rispetto al precedente dal punto di vista del RMSE.

Dato che il modello polinomiale richiede in input due parametri (regolarizzazione e ordine del polinomio) decidiamo di implementare una GridSearch, ottenendo così diverse opzioni tra cui scegliere. Decidiamo di settare come ordine del polinomio valori compresi tra il 3 e il 7 (escludiamo l'1 in quanto sarebbe come generare un modello lineare, il 2 perché una parabola non fa al caso nostro, e settiamo a 7 il valore massimo in modo tale da rischiare di non ottenere un overfitting del modello), mentre per il parametro di regolarizzazione utilizziamo dei valori pari a 0.001, 0.01, 0.1, 1 e 1.5. Dopo aver plottato i vari risultati notiamo come il RMSE è migliore con i modelli con ordine del polinomio maggiore, come ci aspettavamo, tuttavia notiamo come il modello con ordine 7 preveda in maniera errata dopo il punto di minimo in x = 5 circa, in particolare sembra che la curva delle previsioni abbia una pendenza maggiore rispetto a quella dei valori reali, quindi optiamo per un polinomio di grado 6. Per quanto riguarda il parametro di regolarizzazione decidiamo di utilizzare un valore di 0.001, in quanto minimizza (seppur di poco) l'errore.



Gaussian Ridge Regression

Proviamo infine ad utilizzare un modello con un kernel gaussiano per verificare se è possibile migliorare ultimamente il modello. Analogamente al modello polinomiale, il modello gaussiano richiede in input due parametri, tuttavia in questo caso il parametro di regolarizzazione è fissato a 0.1, quindi è necessario solamente verificare per quale valore di gamma si ottiene un modello migliore. Dopo aver scelto i valori di gamma (0.001, 0.01, 0.1, 1, 2 e 3) procediamo col plottare i vari modelli e notiamo come il modello con gamma = 1 registra un RSME minore.

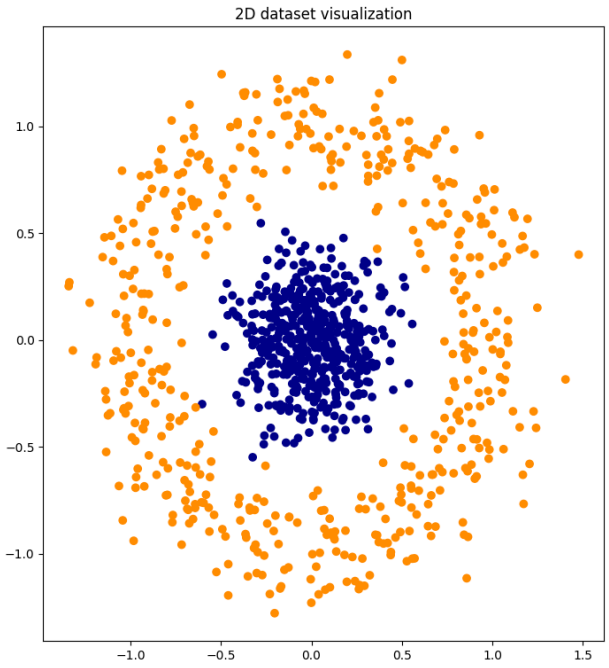


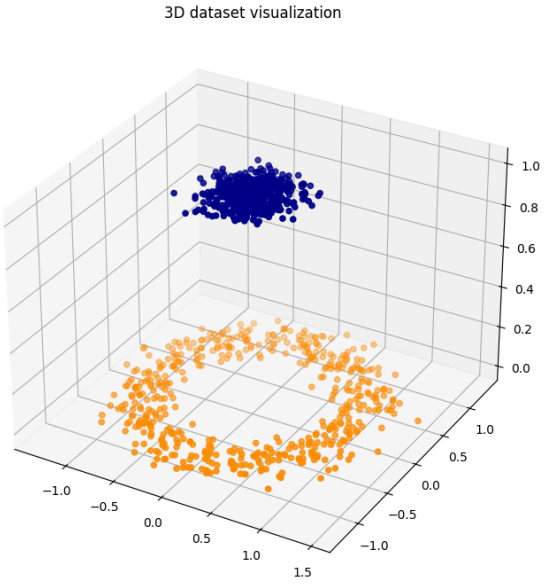
Possiamo quindi concludere dicendo che il modello che fitta al meglio il nostro dataset è il modello di Ridge Regression con kernel gaussiano con un gamma = 1.

PCA

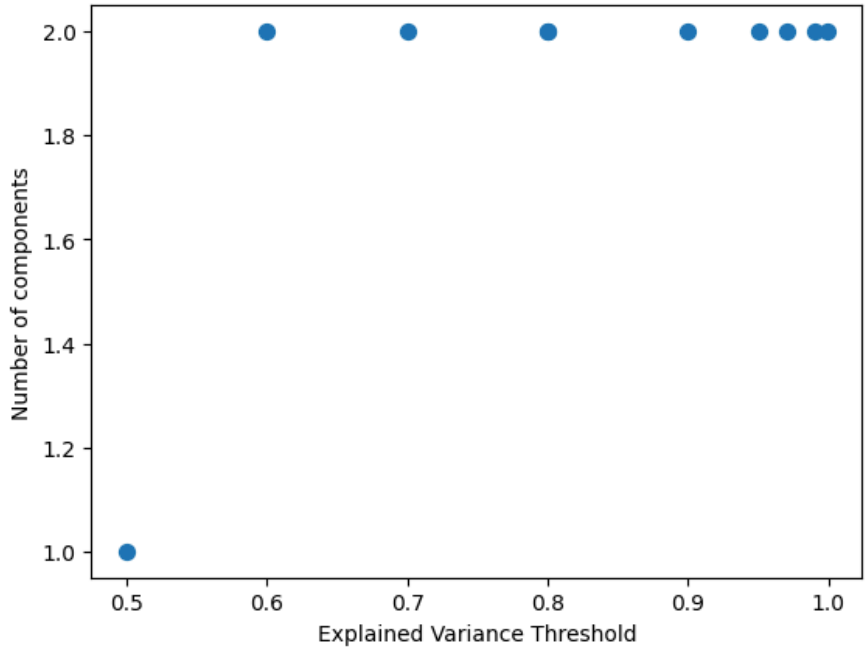
Per quanto riguarda la PCA i dati sono stati generati utilizzando la funzione make_circles di sklearn, e abbiamo subito verificato se ci fosse qualche tipo di ordinamento nelle labels, in modo da poter eventualmente applicare qualche funzione di shuffling per evitare di trovarsi ad avere un training set ed un test set "sbilanciati" sotto questo aspetto (es. avere tutte y = 0 nel test set). Siccome non è

sembrato esserci alcun tipo di ordinamento abbiamo proceduto con il plot del nostro dataset, prima in 2 dimensioni, e successivamente in 3.

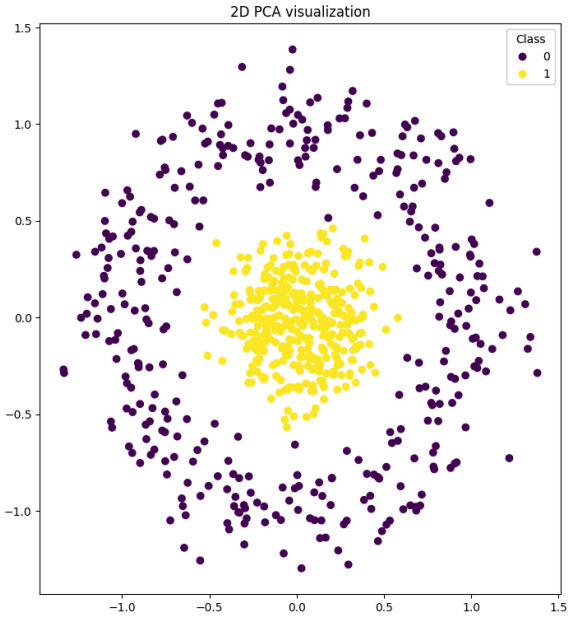


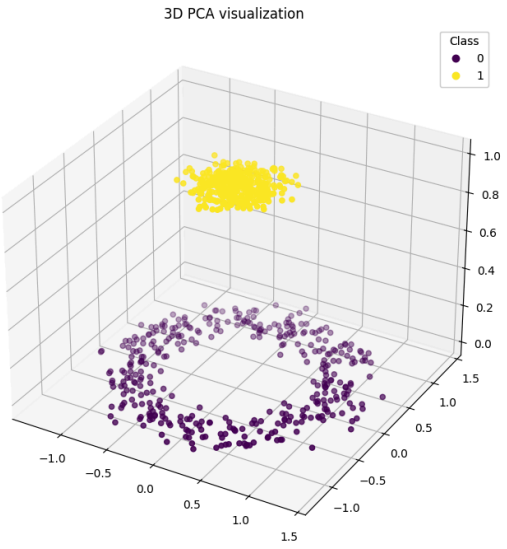


Solitamente PCA viene utilizzato per ridurre la dimensionalità, e quindi utilizzare un determinato numero di features significative in base alla loro varianza spiegata, ma in questo caso il numero di features è due, quindi l'unica alternativa ad usarle entrambe è usarne solo una, così abbiamo verificato quale fosse l'effettiva varianza spiegata in base al numero di features, e questo è ciò che abbiamo ottenuto:



Si può notare immediatamente come, utilizzando una sola feature, non si riesca ad ottenere un valore alto, perciò abbiamo continuato ad utilizzarle entrambe, ottenendo così i seguenti grafici:





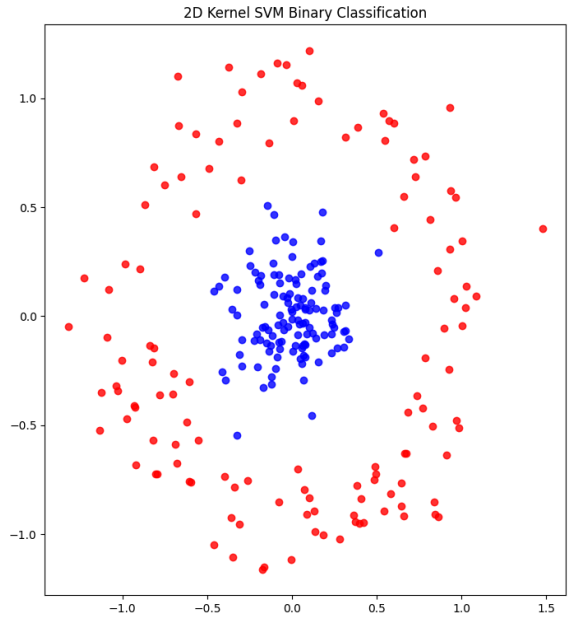
È facile notare come la rappresentazione grafica ottenuta tramite la PCA e la visualizzazione del dataset iniziale siano pressoché simili, in quanto la dimensionalità del dataset è rimasta invariata.

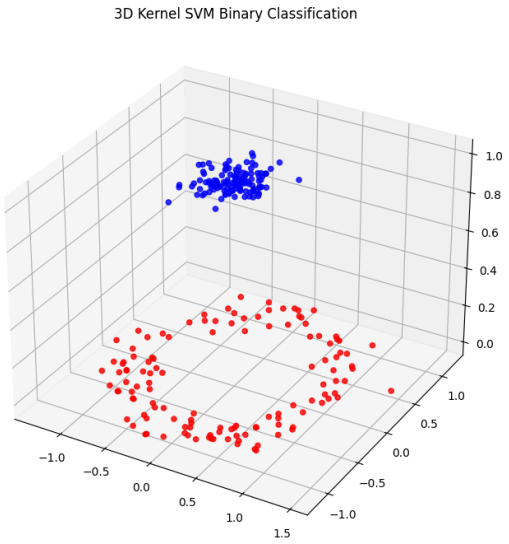
RBF Kernel PCA

Dalla PCA emerge come vi sia una effettiva separazione tra le due classi, ed è più chiaro se si guarda la rappresentazione 3D di come ci possa essere un iperpiano a separarle, così abbiamo provato ad ottenere un risultato migliore utilizzando un RBF kernel.

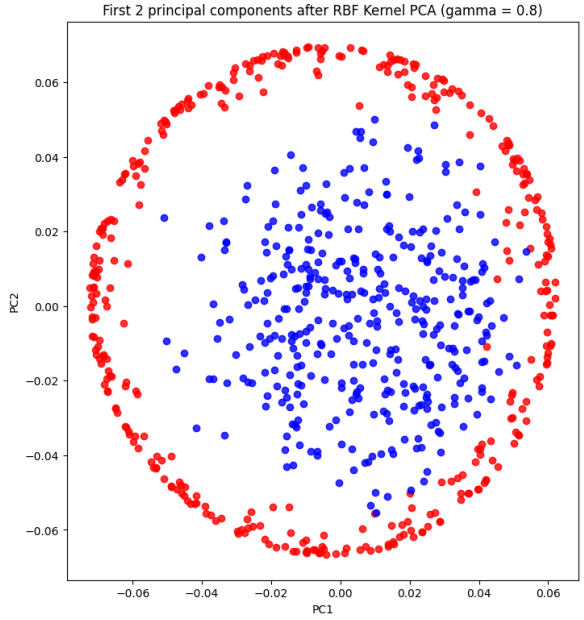
Implementando la tecnica SVM (Support Vector Machine), abbiamo verificato come si comportasse il nostro problema di classificazione binaria sul test set. Inoltre, anche nella PCA ci siamo serviti della grid search per determinare quale fosse il miglior parametro da utilizzare, il quale è risultato essere gamma = 0.8. Abbiamo quindi applicato SVM utilizzando come parametro gamma = 0.8, ottenendo così una accuracy score pari a 0.996, un risultato ottimo.

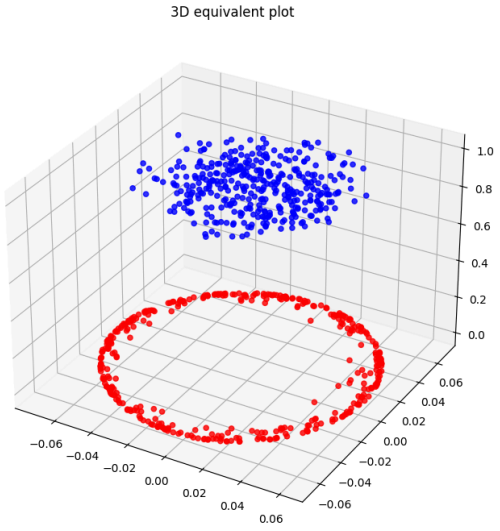
Successivamente abbiamo plottato il dataset ottenuto dopo aver applicato la classificazione mediante SVM con KRB.





Infine abbiamo plottato le 2 componenti principali ottenute tramite la PCA utilizzando il kernel RBF.





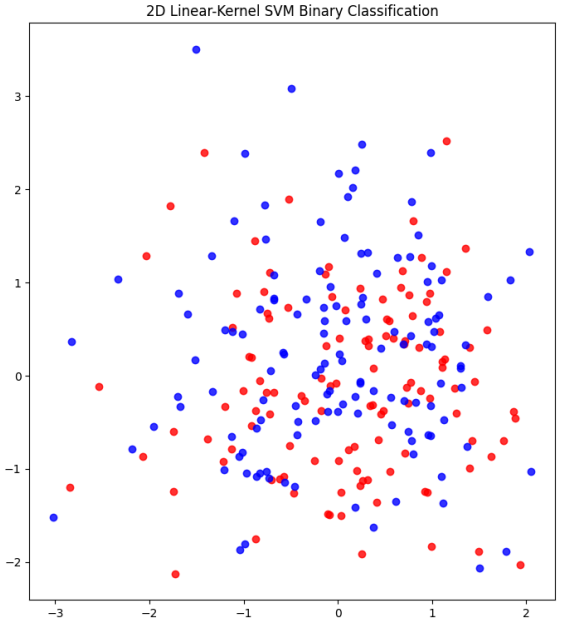
Si può notare come l'effetto principale derivante dall'uso del RBF kernel sia la presenza di alcuni punti colorati di rosso, che geometricamente apparterrebbero al cluster dei punti blu guardando il grafico bidimensionale, ma che vengono colorati diversamente in quanto classificati correttamente. Inoltre, in quanto KPCA applica il

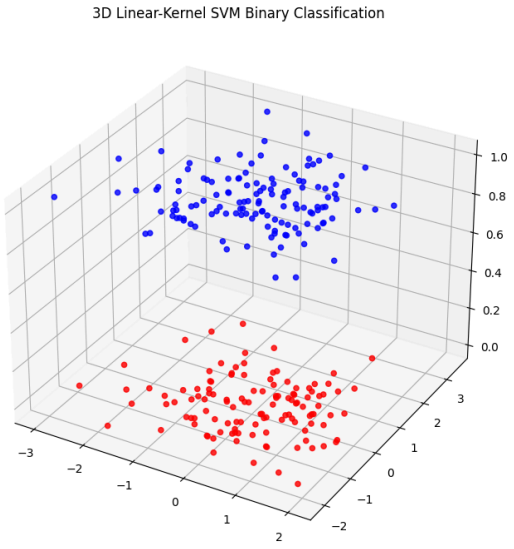
kernel ai dati prima di effettuare la PCA, si può distinguere un chiaro pattern geometrico formato dai punti rossi.

PCA pt.2

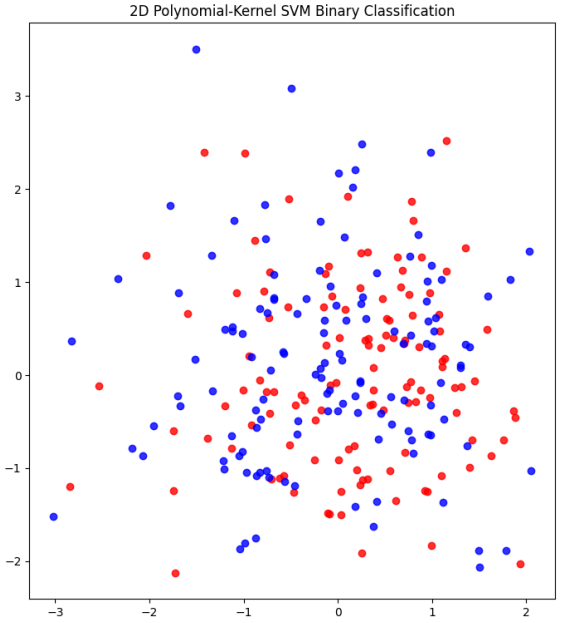
Abbiamo infine ripetuto i procedimenti soprastanti per il dataset generato dalla funzione “datasets.make_classification”, e lo abbiamo utilizzato per testare 3 diversi tipi di kernel, dei quali riportiamo il nome, le relative accuracy scores ottenute ed i plot ricavati dall’utilizzo del relativo kernel.

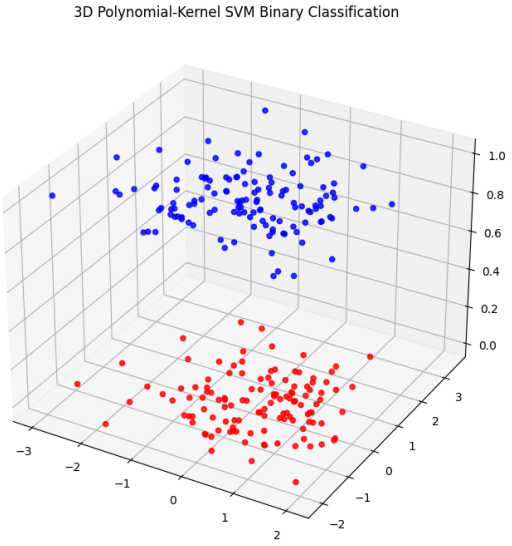
- Linear Kernel: accuracy score di 0.876



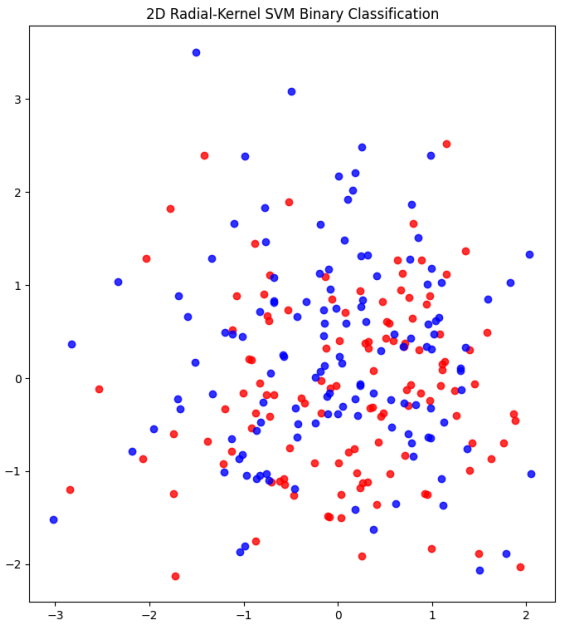


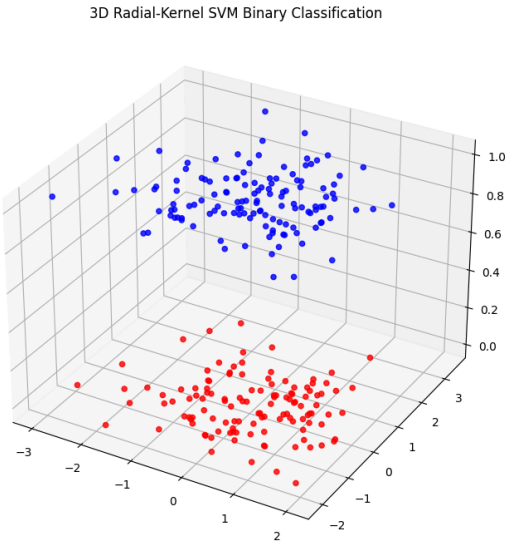
- Polynomial Kernel: accuracy score di 0.796





- RBF Kernel: accuracy score di 0.892





L'aspetto più significativo degno di nota che si osserva dai plot soprastanti è che in tutte le tecniche utilizzate i punti sono disposti in maniera pressoché uguale. Possiamo quindi dedurre che le varie trasformazioni apportate dai kernel hanno un impatto limitato sulla struttura dei dati rispetto alla riduzione dimensionale fornita dalla PCA. Questo risultato è ulteriormente confermato dalle accuracy scores ottenute dai vari modelli, in quanto si aggirano tutte intorno allo stesso range. In

conclusione, possiamo affermare che il kernel migliore da utilizzare sia quello RBF, in quanto permette di ottenere un’accuratezza maggiore rispetto agli altri.