Appendix

Proof of learning algorithm

If we add an extra mask layer in the base model, the feedforward propagation becomes :

$$\begin{cases} a_j^{l+1} = a_j^l w_j^{l,l+1} \text{ if } l = 0 \\ a_j^{l+1} = \sigma(\sum_i a_i^l w_{ij}^{l,l+1} + b_j^{l+1}) \text{ if } l > 0 \end{cases} \quad (1)$$

where w_{ij}^k is weight for node j in layer l_k for incoming node i; σ is the activation function; b_j^k is bias for node j in layer l_k and a_j^l is product sum plus bias (activation) j in layer l_k.

An error function $E(X, \theta)$ defines error between the desired output y and the calculated output $\hat{y}$ of the neural network with the parameter θ with respect to the weights w_{ij}^k and biases b_j^k. In the training process, weights are updated according to the gradient descent of each iteration. Based on the learning rate η, each iteration of gradient descent updates as,

$$\theta^{t+1} = \theta^t - \eta \frac{\partial E(X, \theta^t)}{\partial \theta} \quad (2)$$

where θ^t is parameters of the network at iteration t. To illustrate the calculation details of the model with a mask layer the mean squared error function is selected, defined as

$$E(X, \theta) = \frac{1}{2N} \sum_{i=1}^{N} (\hat{y} - y)^2 \quad (3)$$

According to the backpropagation algorithm and chain rule, the derivation of gradients is straightforward. By deriving $\frac{\partial E(X,\theta)}{\partial w_{ij}^k} = \frac{1}{N} \sum_{d=1}^{N} \frac{\partial E_d}{\partial w_{ij}^k}$, the general form for all input in X can be summarized by adding all individual gradients. The individual gradient with respect to a general weight is $\frac{\partial E_d}{\partial w_{ij}^k}$, where the chain rule applies. The error function partial derivative becomes $\frac{\partial E_d}{\partial \alpha_j^k} \frac{\partial \alpha_j^k}{\partial w_{ij}^k}$. The first term is usually called the error, denoted δ_j^k, and the second term can be simplified as o_i^{k-1}. The first term can be transferred to the following equation by applying the chain rule:

$$\delta_j^k = \frac{\partial E_d}{\partial \alpha_j^k} = \sum_{l=1}^{r^{k+1}} \frac{\partial E_d}{\partial \alpha_l^{k+1}} \frac{\partial \alpha_l^{k+1}}{\partial \alpha_j^k} = \sum_{l=1}^{r^{k+1}} \delta_l^{k+1} \frac{\partial \alpha_l^{k+1}}{\partial \alpha_j^k} \quad (4)$$

where r^{k+1} is the number of nodes in the next layer. Because all the weights (except for those between the input layer and the mask layer) are not trainable, they can be considered as constants.

According to the linear relationship shown in Fig.1, the partial derivative between the input layer and the mask layer can be derived as

$$\frac{\partial E}{\partial \alpha_j^1} = \delta_j^1 o_i^0 \delta_j^1 = \sum_{l=1}^{r^2} \delta_j^2 \frac{\partial \alpha_l^2}{\partial \alpha_j^1} = \sum_{l=1}^{r^2} \delta_j^2 w_{jl}^2 \frac{\partial E}{\partial \alpha_j^1} = o_i^0 \sum_{l=1}^{r^2} \delta_j^2 w_{jl}^2 \quad (5)$$

where o_i^0 is input data directly from X. In this equation Fig.5, we can observe that o_i^0 and w_{jl}^2 are constant and the only variable is δ_j^2. If we trace this down to the final layer δ_j^m, in which m is the final layer and the formula defined as $(\hat{y} - y)g'(\alpha_j^m)$, where g is the activation function, the only factor that affects the weights between the input layer and the mask layer is the prediction error. Thus, if a feature is important to the outcome, causing the substantial change in prediction error, the coefficient (weight in the model) of the feature would be updated accordingly. However, the coefficient (weight in the model) of unimportant features will not deviate significantly from the initialization value. Overall, the θ accumulates the effect of all connection weights across all hidden neurons.

The effect of the prediction error $(\hat{y} - y)$ is same in each iteration to all neurons in the mask layer and the factor that controls the θ is the weight w_{jl}^2 in Fig.5 obtained from the base model. Considering that the weight w_{jl}^2 is calculated based on all connection weights between its connected neurons in hidden layers, the weights imply the importance of the variance.

Derive of Contribution Factor

Considering the contribution of each feature in the feature set and the linear relationship between mask and feature set, we can derive $\mathbb{C}(Q, (X_0, X_1, X_2, ..., X_i)) = \mathbb{C}(Q, (w_{X_0} 0 X_0, w_{X_1} X_1, w_{X_2} X_2, ..., w_{X_i} X_i))$. As the model is fixed, the function can be simplified into the following function.

$$\sum_{i=0}^{N} \mathbb{C}(Q, X_i) = \sum_{i=0}^{N} \mathbb{C}(Q, w_{X_i} X_i) \quad (6)$$

To solve the equation with unknowns, the exact corresponding number of equations have to be provided, offered by retraining the system. The w_{X_i} in the function also needs to be moved outside the parentheses, where the w_{X_i} represents the relationship between the weight and the contribution.

Linear regression case explanation Thinking of the most straightforward model, a linear regressor, the coefficient of a feature is linear to its contribution, as increasing the weight by one unit changes the estimated outcome by 100%, which doubles the contribution of the feature. So the following function can be derived: $\mathbb{C}(X_1) + \mathbb{C}(X_2) + ... + \mathbb{C}(X_i) = w_{X_1}\mathbb{C}(X_1) + w_{X_2}\mathbb{C}(X_2) + ... + w_{X_i}\mathbb{C}(X_i)$ from Eqn.6. If the equation system can be solved, then the contribution of each feature to the model performance can be decided.

Fortunately, the previous retrain strategy provides a chance out! Given the number of unknown parameters, we can solve the equation system using the exact corresponding number of equations, offered by retraining the system. The following equation system can be solved easily. By setting the output of each equation to 1, the relationship among all features can be derived. The system can be solved using its associated augmented matrix, as it is empirically more suitable for computer manipulations $(A|I) = \begin{pmatrix} 1 & 1 & ... & 1 & | & 1 \\ w_{X_1} & w_{X_2} & ... & w_{X_i} & | & 1 \\ w'_{X_1} & w'_{X_2} & ... & w'_{X_i} & | & 1 \end{pmatrix}$.

$$\begin{cases} \mathbb{C}(X_1) + \mathbb{C}(X_2) + ... + \mathbb{C}(X_i) = 1 \\ w_{X_1}\mathbb{C}(X_1) + w_{X_2}\mathbb{C}(X_2) + ... + w_{X_i}\mathbb{C}(X_i) = 1 \\ \vdots \\ w'_{X_1}\mathbb{C}(X_1) + w'_{X_2}\mathbb{C}(X_2) + ... + w'_{X_i}\mathbb{C}(X_i) = 1 \end{cases} \tag{7}$$

Introduction and derive of μ However, in practice the model usually is much more complex and the factor w_{X_i} can not be applied to the contribution directly. To deal with the coefficient in general cases, the model can be run with different input settings to determine the relationship between function $(\cdot)$ and w by μ experimentally.

For a specific coefficient w_{X_0} of a feature X_0, all other coefficients are fixed except for the one that is being calculated and we compare the performance between the model with input $\chi(X_0, X_1, X_2, ..., X_i)$ and $\chi(w_{X_0}X_0, X_1, X_2, ..., X_i)$ by multiplying a mask $(w_{X_0}, 1, ..., 1)$ to χ. The change in ΔP between these two outcomes $\Delta P_{partial} = \mathbb{P}(\mathbb{Q}(X_0, X_1, ..., X_i)) - \mathbb{P}(\mathbb{Q}(w_{X_0}X_0, X_1, ..., X_i))$ is from $X_0 \mapsto w_{X_0}X_0$ and represents the effect of part of the X_0. The total contribution of feature X_0 can be found by setting $\Delta P_{total} = \mathbb{P}(\mathbb{Q}(X_0, X_1, ..., X_i)) - \mathbb{P}(\mathbb{Q}(0 \times X_0, X_1, ..., X_i))$

According to the $\Delta P_{partial}$ and ΔP_{total} above, the $\mathbb{C}(w_{X_0}X_0)$ can be represented by $\mu \times \mathbb{C}(X_0)$, where the term μ is introduced to represent $\frac{\Delta P_{partial}}{\Delta P_{total}}$ for simplicity, and the simple linear case can be extended to the general case based on Fig.7. The system can then be solved with row operations from Gaussian elimination, including swapping rows positions, add one row onto another and multiplying a row by a non-zero scaler.

Taking a linear regressor as an example, $Q = X_1 + X_2 + X_3$, given an input instance (0.1, 0.3, 0.6), the output is $1 = 0.1 + 0.3 + 0.6$, in which X_1(0.1), X_2(0.3) and X_3(0.6) can be seen providing 10%, 30% and 60% contribution to the model Q. However, the contribution is not always obvious in real cases. In this simple case, to meet the output requirement, the following functions Fig.8 can be defined based on our method above and we can obtain an augmented matrix coefficient matrix $(A|I) = \begin{pmatrix} 1 & 1 & 1 & | & 1 \\ 1 & 2 & 1/2 & | & 1 \\ 4 & 1 & 1/2 & | & 1 \end{pmatrix}$. The matrix gives a relation between features $3 \times \mathbb{C}(X_1) = \mathbb{C}(X_2); 6 \times \mathbb{C}(X_1) = \mathbb{C}(X_3)$, which satisfies the contribution distribution.

$$\begin{cases} \mathbb{C}(X_1) + \mathbb{C}(X_2) + \mathbb{C}(X_3) = 1 \\ \mathbb{C}(X_1) + 2 \times \mathbb{C}(X_2) + 1/2 \times \mathbb{C}(X_i) = 1 \\ 4 \times \mathbb{C}(X_1) + \mathbb{C}(X_2) + 1/2 \times \mathbb{C}(X_i) = 1 \end{cases} \tag{8}$$

Linear and Logistic Regression Supporting Information Supporting information for linear regression and logistic regression, including linear regressor and logistic regressor training settings Table.2, linear and logistic base model and feature model structures Table.3, the training settings of independent model Table.5, learning curves of the base model (left column) and the feature model (right column) Fig.2, weights distribution of linear and logistic models Fig.3, profile change of linear regression and logistic regression as the number of iterations increases Fig.4, the feature importance profile of Boston house price and Breast Cancer Wisconsin from different feature selection methods 7, important features and detailed contribution of each feature Fig.5 and CF of linear regression and logistic regression with equations accumulated Fig.6.

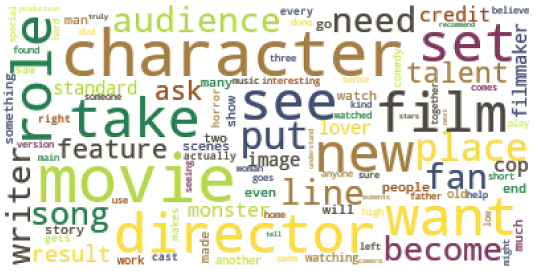

Figure 1: Unimportant words.

Application Supporting Information

Supporting information for applications, including RNN base model and feature model structures Fig.6, RNN model training settings Table.8, Gold training settings Table.8, results of 5 models trained on different Gold feature sets Table.11, Multilayer perceptron base model and feature model structures Table.1, CNN base model and feature model structures Table.7, CNN model training settings Table.10 and feature unimportance maps from different methods Fig.8.

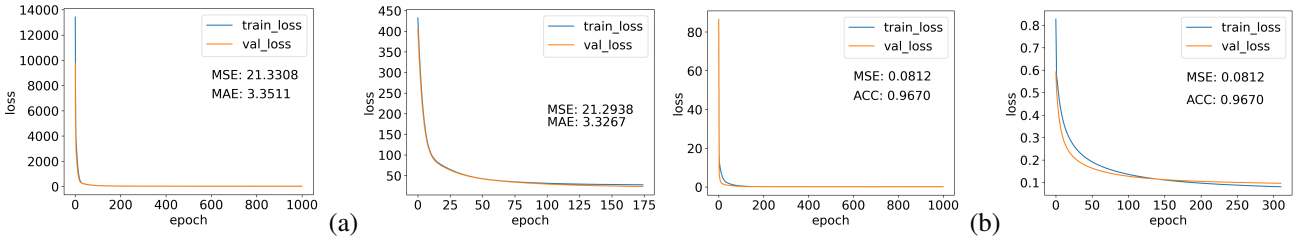

Figure 2: The learning curve with results of the base model (left column) predicting the target property and the feature model (right column) trained to achieve the same performance, showing the accuracy and generalizablity. (a) Linear regression, (b) Logistic regression

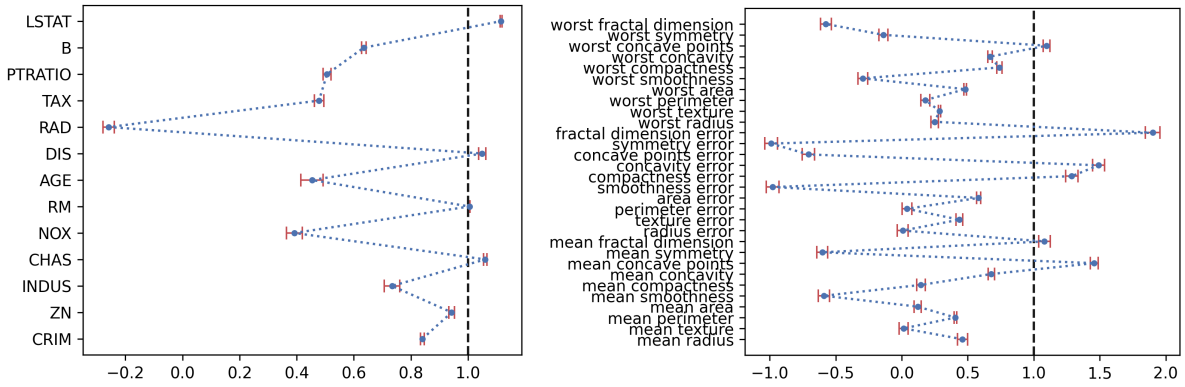

Figure 3: The weights of linear regression and logistic regression are displayed in weight plot and the target 1 is marked as the line

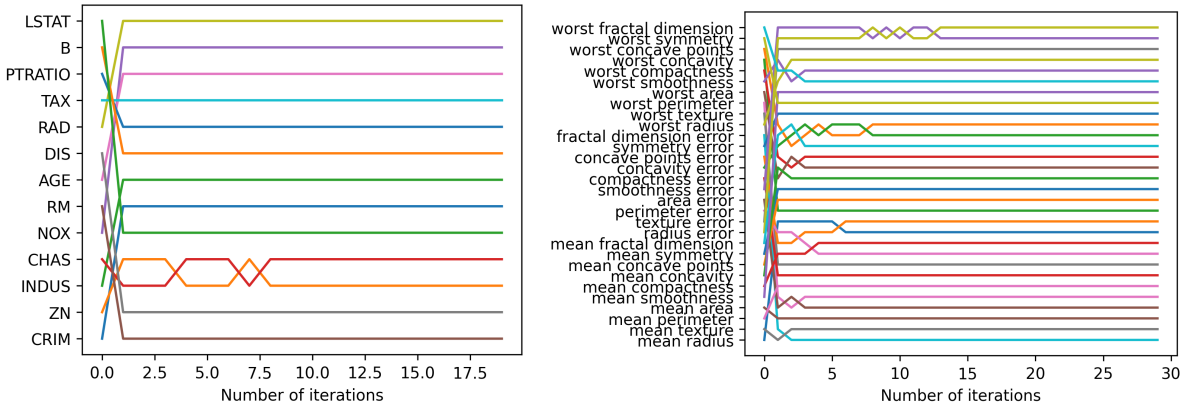

Figure 4: Profile change of linear regression and logistic regression as the number of iterations increases

	Bulk	Surface	Total	Condensed	Expanded
MAE	0.1109	0.1355	0.0981	0.0615	0.0529
MSE	0.0253	0.0421	0.0213	0.0087	0.0117

Table 1: Results of 5 linear models trained on Bulk (B), Surface (S), Totals (T), Condensed (C) and Expanded (E), Gold feature sets

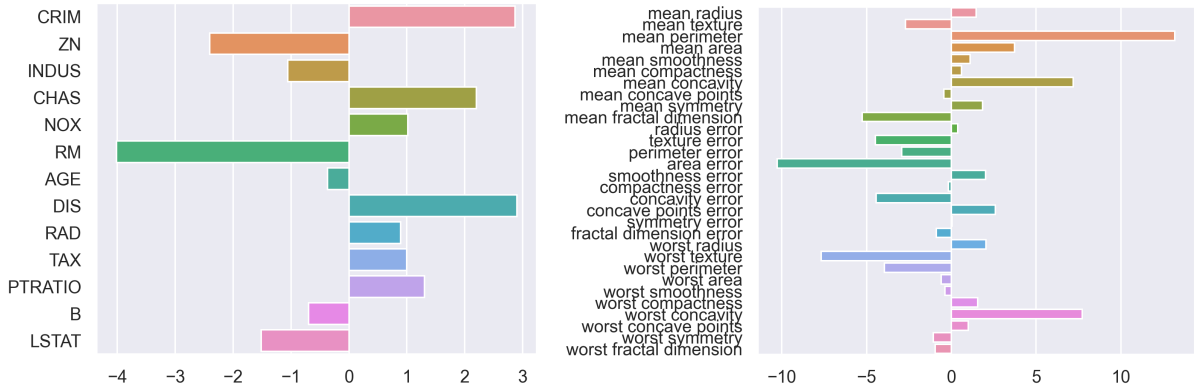

Figure 5: Detailed contribution of each feature from CF.

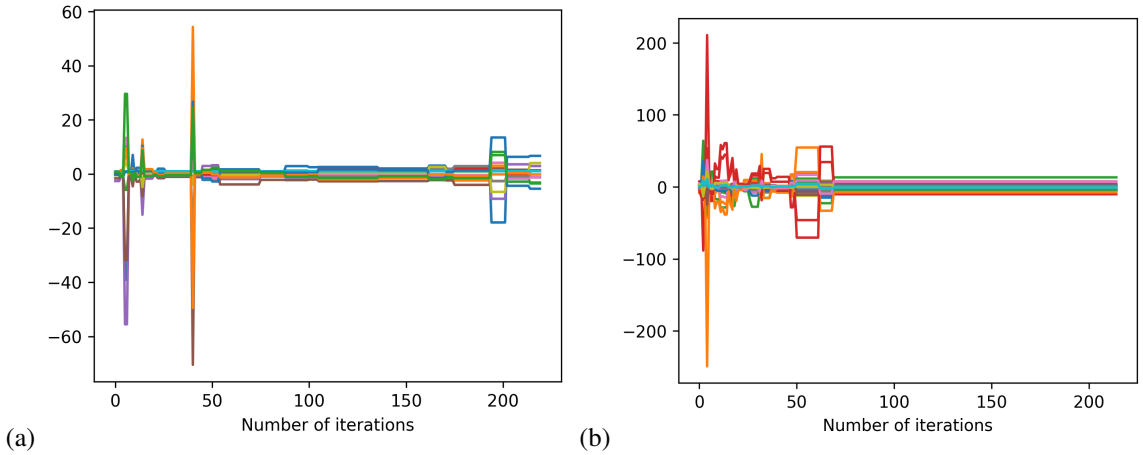

Figure 6: Contribution distribution of linear regression and logistic regression with equations accumulated

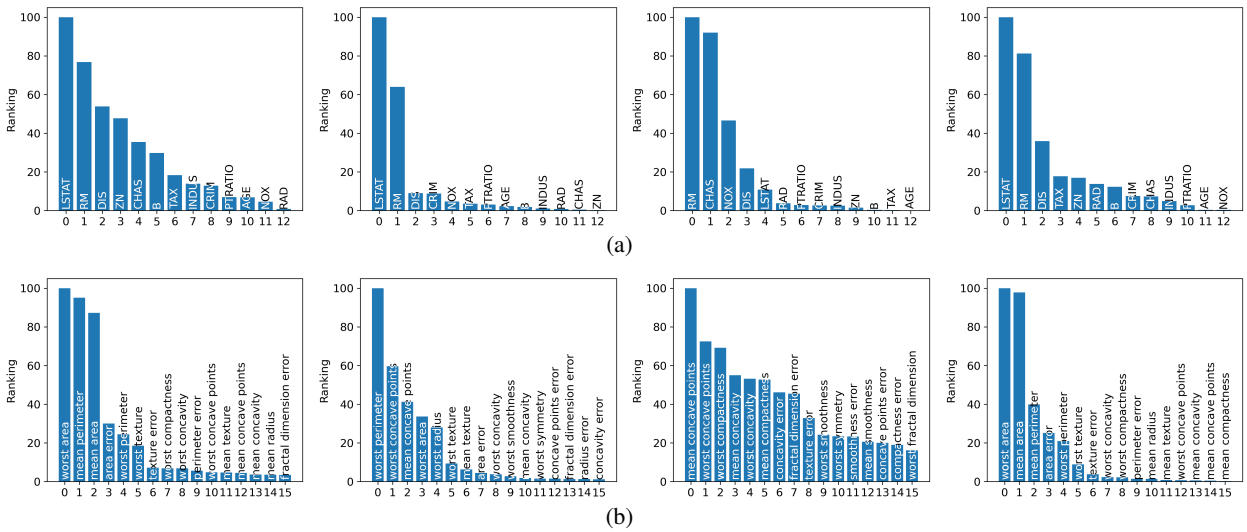

Figure 7: The feature importance profile of Boston house price and Breast Cancer Wisconsin from KernalSHAP, random forest (RF), Connection weights, Permutation method (from left to right) (a) Linear regression, (b) Logistic regression

	Dataset	Model	Optimiser	Epoch	Batch size	Loss
Base model	Boston house price	Linear regressor	ADAM	1000	10	MSE
Feature model	Boston house price	Linear regressor	ADAM	-	10	MSE
Base model	Breast Cancer Wisconsin Dataset	Logistic classifier	ADAM	1000	10	binary_crossentropy
Feature model	Breast Cancer Wisconsin Dataset	Logistic classifier	ADAM	-	10	binary_crossentropy

Table 2: Linear regressor and logistic regressor training settings (seed is 3)

	Layer	Layer type	Layer parameters	Num of trainable parameters	Output size
Base model	L0	Input layer	-	14	N, 13
	L1	Dense layer	-	14	13, 13
	L2	Output layer	-	14	None, 1
Feature model	L0	Input layer	-	0	N, 13
	L1	Mask layer	-	13	13, 13
	L2	Dense layer	-	13	13, 13
	L3	Output layer	-	13	None, 1

Table 3: Linear base model and feature model structures

	Layer	Layer type	Layer parameters	Num of trainable parameters	Output size
Base model	L0	Input layer	-	31	N, 30
	L1	Dense layer	sigmoid	31	30, 30
	L2	Output layer	-	31	30, 1
Feature model	L0	Input layer	-	30	N, 30
	L1	Mask layer	-	30	30, 30
	L2	Dense layer	sigmoid	30	30, 30
	L3	Output layer	-	30	30, 1

Table 4: Logistic base model and feature model structures

Dataset	Model	Optimiser	Epoch	Batch size	Loss
Boston house price	ANN	ADAM	150	100	MSE
Breast Cancer Wisconsin Dataset	ANN	ADAM	150	100	binary_crossentropy
MNIST	CNN	ADAM	150	100	binary_crossentropy

Table 5: Independent model training settings (seed is 3)

Base Model structure	Layer	Layer type	Layer parameters	Num of trainable parameters	Output size
	L0	Embedding layer	-	64000	None, 200, 32
	L1	LSTM layer	-	53200	None, 100
	L2	Dense layer	sigmoid	101	None, 1
Feature Model structure	L0	Onehot layer	-	0	None, 200, 2000
	L1	Mask layer	-	2,000	None, 200, 32
	L2	LSTM layer	-	53200	None, 100
	L3	Dense layer	sigmoid	101	None, 1

Table 6: RNN base model and feature model structures

Base Model structure	Layer	Layer type	Layer parameters	Pool size	Kernel size	Num of trainable parameters	Output size
	L0	Input layer	-	-	-	-	None, 28, 28, 1
	L1	Convolutional layer	Relu	-	3x3	640	None, 26, 26, 64
	L2	Max pooling layer	-	2x2	-	0	None, 13, 13, 64
	L3	Convolutional layer	Relu	-	3x3	18464	None, 11, 11, 32
	L4	Max pooling layer	-	2x2	-	0	None, 5, 5, 32
	L5	Flatten layer	-	-	-	-	None, 800
	L6	Drop out layer	0.5	-	-	0	None, 800
	L7	Dense layer	Softmax	-	-	1602	None, 2
Feature Model structure	L0	Input layer	-	-	-	-	None, 28, 28, 1
	L1	Mask layer	-	-	-	784	None, 28, 28, 1
	L2	Convolutional layer	Relu	-	3x3	0	None, 26, 26, 64
	L3	Max pooling layer	-	2x2	-	0	None, 13, 13, 64
	L4	Convolutional layer	Relu	-	3x3	0	None, 11, 11, 32
	L5	Max pooling layer	-	2x2	-	0	None, 5, 5, 32
	L6	Flatten layer	-	-	-	-	None, 800
	L7	Drop out layer	0.5	-	-	0	None, 800
	L8	Dense layer	Softmax	-	-	0	None, 2

Table 7: CNN base model and feature model structures

	Dataset	Model	Optimiser	Epoch	Batch size	Loss
Base model	IMDB	RNN	ADAM	500	64	binary_crossentropy
Feature model	IMDB	RNN	ADAM	-	64	binary_crossentropy

Table 8: RNN model training settings

Base Model structure	Layer	Layer type	Layer parameters	Num of trainable parameters	Output size
	L0	Input layer	-	-	N, 10
	L1	Dense layer	-	1728	10, 32
	L2	Dense layer	-	528	10, 16
	L3	Output layer	-	17	10, 1
Feature Model structure	L0	Input layer	-	0	N, 53
	L1	Mask layer	-	53	N, 53
	L3	Output layer	-	53	10, 1

Table 9: Multilayer perceptron base model and feature model structures

	Dataset	Model	Optimiser	Epoch	Batch size	Loss
Base model	MNIST	CNN	ADAM	1000	128	binary_crossentropy
Feature model	MNIST	CNN	ADAM	-	128	binary_crossentropy

Table 10: CNN model training settings

	Dataset	Model	Optimiser	Epoch	Batch size	Loss
Base model	Gold nanoparticles	Linear regressor	ADAM	1000	10	Mse
Feature model	Gold nanoparticles	Linear regressor	ADAM	-	10	Mse

Table 11: Gold training settings

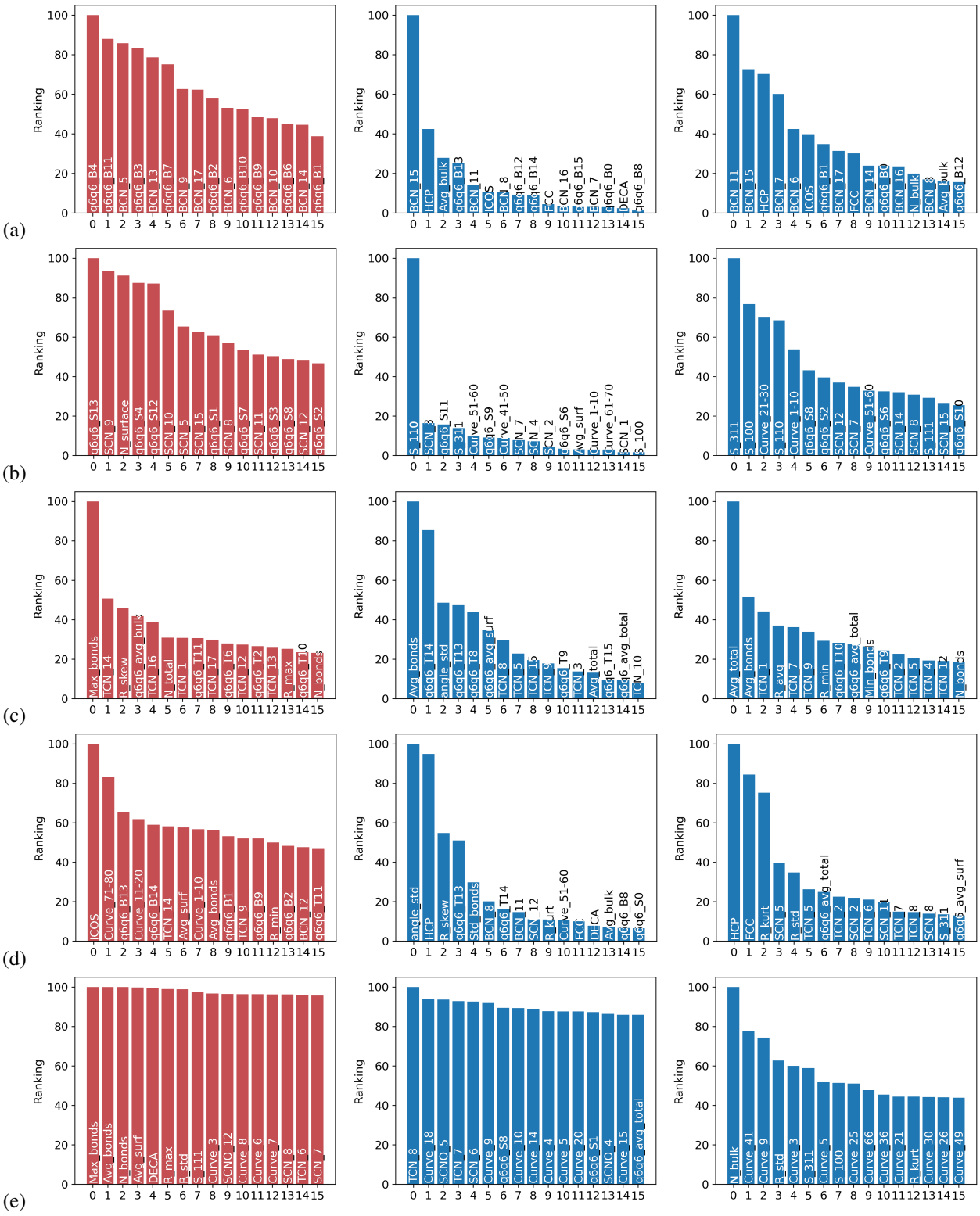

Figure 8: Feature unimportance maps (left) from VTF and feature importance maps, including RVTW and CF (middle and right) of Gold feature sets on different feature sets. (a) Bulk, (b) Surface, (c) Total, (d) Condensed, (e) Expanded.