SUPPLEMENTARY DATA

Materials and Methods

Feature Engineering

The initial dataset of 39,756 maize B73v5 genes had 874 missing values for the target label core/non-core. As a result, we excluded the 874 genes from the original dataset and kept them as supplementary test data to categorize them after model development while using the remaining dataset of 38,882 genes for our subsequent analysis (Supplementary Figure S8). Among the 38,882 genes, seventy-two percent of our genes marked as core in the maize reference genome version B73v5 were considered as the positive sample, and twenty-eight percent annotated as non-core ( near-core, dispensable and private genes) were considered as the negative sample for modeling. This dataset was then transformed by substituting zeros for missing values and one-hot encoding the categorical variables. In one hot encoding process, categorical variables are converted into a form that ML algorithms could use to perform better in prediction.

Handling Data imbalance

Rather than using random down-sampling methods, as had been previously discussed in the data visualization phase, the modeling phase handled data imbalance issues using the synthetic minority oversampling technique (SMOTE). Downsampling based on the minority class may lead to data loss since the majority class is automatically downsampled based on the minority class, and data loss is expensive for modeling. On the other hand, SMOTE is a technique for oversampling in which synthetic samples are created for the minority class. It is critical to emphasize that SMOTE resampling was conducted exclusively on training data, because if we did SMOTE on the entire dataset and then split it, SMOTE would pass some information into the testing set, thereby skewing the results in an optimistic direction.

Hybrid Feature Selection

A total of 14,405 characteristics were discovered for all previously annotated genes. This magnitude may provide significant challenges for many machine-learning algorithms with scalability and learning performance. For instance, high-dimensional data (datasets with hundreds or thousands of characteristics) may contain a high proportion of irrelevant and redundant information, significantly impairing the performance of learning algorithms. As a result, feature selection becomes critical for machine-learning tasks when confronted with high-dimensional data such as ours. We used a hybrid feature selection model in both the “Advanced” and “Basic” models. Here we discuss the hybrid feature selection technique that is used for the "Advanced" model, but a similar approach is used for the "Basic" model. After excluding features with nearly 80% missing values, we eliminated highly correlated features by setting a threshold of 0.8 and discarding columns with correlations more than that threshold; hence, 13,300 characteristics (per gene) were preserved and used in further analyses. From here, two rounds of feature selection were performed (Supplementary Figure S9). The first phase is technically referred to as “feature selection by model”, while the second phase is referred to as "recursive feature elimination with cross-validation" (RFECV). The recursive feature elimination with cross-validation is a greedy optimization algorithm. It is more accurate than feature selection by model but is computationally very expensive, therefore the hybrid approach uses feature selection by model to reduce the computational complexity of wrapper algorithms (recursive feature elimination with cross-validation).

Feature selection by model

We applied four different machine-learning models for feature selection: L1-based linear regression, Extremely Randomized Trees (Extra-trees model), Random Forest, and Gradient Boosting classifiers. We evaluated each model's performance on the validation data and plotted the model’s importance rankings for each model. Because the Gradient Boosting Classifier achieved the best and most balanced accuracy in both the training and validation sets, we use it to choose the top 100 features (from Supplementary Figure 10A) before the feature importance values saturate.

Recursive Feature Elimination (RFE) with cross-validation

The second phase of feature selection involves determining the optimal combination of features. We accomplish this through "Recursive Feature Elimination" (RFE). Rather than creating a single model, we construct *n* Gradient Boosting models (where *n* equals the number of features). We train the model using all 100 features in the first iteration and calculate the 5-fold cross-validation area under the Receiver Operating Characteristic Curve (AUC-ROC) score (an evaluation metric that plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold values) and the feature importance of each column. Then we eliminate the least important feature, leaving us with 99. We repeat the process above based on these 99 features until we reach the final single feature. This method is time-consuming but creates a reliable feature importance ranking.

The graph (Supplementary Figure 10B) clearly shows that the validation AUC-ROC score will eventually saturate when trained with more than 40 features (around 0.97). As a result, we end up retaining the 40 most critical characteristics. To take advantage of other models' embedded feature selection techniques, we repeat the procedure with the other three models stated earlier and combine the prediction of multiple weak learners to form a strong one. Combining all these steps and using multiple Sklearn models, we created a feature selector that retains the 98 most important features for subsequent analysis. However, our objective is to implement the predictive model in a real-world setting, further reducing the number of features, as fewer variables imply fewer user input requirements. This also decreases the likelihood of error while filling out the input form for modeling. Therefore, we chose the top 25 features out of the 98 most important features to carry out our hyperparameter tuning and deployment. The base Gradient boosting classifiers was used to rank the 98 most important features and select the top 25 features.

Model building and validation

We cover the model building and validation step for the "Advanced" model to evaluate the  
performance of the top features selected during the feature selection step. The "Basic" model  
follows a similar strategy but uses a different set of features (Supplementary Table S3). The union selector step in our “Hybrid Feature selection “ resulted in 98 most important features. In union selector, we applied a number of recursive feature elimination methods with 5-fold cross-validation to pick the top features and combine them together by taking a union of each feature subset. The final output is a collection of distinct features obtained from the combination of features. The major advantage of the union method is that it results in higher performance as it  
takes into account every feature that was chosen by a feature selection method, but the feature  
space can grow exponentially if a high number of features are used in the selection process.  
Moreover, our objective is to implement the predictive model in a real-world setting, further  
reducing the number of features, as fewer variables imply fewer user input requirements. This also decreases the likelihood of error while filling out the input form for modeling. From the model performance perspective: fewer variables mean simpler, more interpretable, less over-fitted models. Therefore, we chose the top 25 features out of the 98 most important features to carry out our hyperparameter tuning and deployment. The base Gradient boosting classifiers was used to rank the 98 most important features.

Prior to reducing the 98 most important features to 25 features, we also evaluated the  
performance of the union feature selection method by training the Gradient Boosting classifier  
with the 98 most important features. The cohort of 38,882 genes is randomly divided into 80%  
(learning set) for developing a prediction model and 20% for validating the developed model  
(validation set). Following the application of SMOTE to our training data, we train a base Gradient  
Boosting Classifier using 5-fold cross-validation; the data was first divided into five-folds, four of  
which were used to train the model, and the fifth was used to assess model  
performance/generalizability. We evaluate the model using a set of performance metrics, including  
recall, precision, area under the Receiver Operating Characteristic Curve (AUC-ROC), and area  
under the Precision-Recall Curve (AUC-PR). We classify using traditional performance measures  
based on the confusion table's four components: true positive (TP), false positive (FP), true  
negatives (TN), and false negatives. AUC-ROC curve is a performance measurement for  
classification problems at various threshold settings. ROC is a probability curve, and the Area  
Under the Curve (AUC) measures the ability of a classifier to distinguish between classes and is  
used as a summary of the ROC curve. It tells how much the model can distinguish between classes  
(core/non-core). The higher the AUC, the better the model predicts non-core genes as non-core  
and core genes as the core. A precision-recall curve plots the precision (y-axis) and the recall (x axis) for different thresholds, much like the AUC-ROC curve. The Gradient boosting classifiers  
trained on the top 25 features result in excellent training performance: Accuracy = 0.92, AUC =  
0.97; performance during testing: Accuracy = 0.90, AUC = 0.87 (Supplementary Figure S17).

Hyperparameter Tuning and Deployment

We attempt to tune the hyperparameters of six different models (1. Logistic Regression, 2. Random Forest Classifier, 3. Gradient Boosting Classifier, 4. Extra Trees Classifier, 5. KNeighborsClassifier, and 6. SVM Classifier) using two distinct optimization approaches (1. Random and 2. Grid Search), respectively, via Scikit-Learns RandomizedSearchCV and GridSearchCV methods (Supplementary Figure S19). Using a helper class, we defined a dictionary of models and a dictionary of parameters for each model.

For both “Advanced” prediction and “Basic” prediction, the supplementary files (Supplementary Table S5-S7) summarize the algorithms' highest mean scores for each criterion (accuracy, average precision, f1, precision, recall, and AUC-ROC). Additionally, generated working sheet for all scores using Excel (Supplementary Table S5-S6) for the “Advanced” prediction allowed tracking different hyperparameter combinations and a more accurate comparison of model performance.

Choosing how to deploy predictive models into production is a complicated process; there are numerous ways to manage the predictive models' lifecycle, numerous formats to store them, and multiple ways to deploy them. There are primarily two modes of model deployment: batch and real-time. In this paper, we used Python's pickle library to store our final models and later deployed it in real-time via Flask APIs on an Apache server, which can easily assist in classifying whether a gene is a core or non-core via a graphical user interface.

Model Flexibility

In each modeling form ("Advanced" or "Basic"), users enter the necessary information for their gene of interest to classify it as core or non-core. Prediction results are displayed at the bottom of the same form with a probability score between 0 and 1. While the "Advanced" model is more accurate and more efficient, the input features required by the "Advanced" model are specific to maize, or use less readily available experimental features. Therefore, the "Advanced" model may only work on maize genes from well-annotated genomes. In addition, the generation of some of these features is time-consuming and requires programming skills. Therefore, to provide a highly versatile prediction platform for both novices and experts, we developed the "Basic" model that relies only on gene and protein sequences and structural features. These genomic features are easily accessible or readily available in the form of GFF files. For users lacking the necessary sequence features, the “Basic” model also includes an input box that will auto-fill the form by taking in only the protein sequence and the coding sequence from the user. Because of this, the “Basic” model acts as a hassle-free prediction platform for a wide range of users.

Additionally, users can retrain both the “Basic” and the “Advanced” models with data specific to their species of interest for better species-specific gene annotation prediction. Users who want to carry out a quick classification of their genes as core or non-core can opt for the “Basic” model, or if the accuracy of the model is critical to them, they can opt for the “Advanced” model.

Application Reproducibility

The Maize Feature Store is a Python application that utilizes the Flask framework and the concept  
of modular programming. We employed a modular software design to make MFS easily readable,  
stable, and maintainable. The application is constructed by segmenting the code. These  
components are referred to as modules. The goal of this separation was to have modules that were  
self-contained or had only a few dependencies on other modules. In other words, the objective was  
to minimize dependency. MFS is a modular system built on top of a customized Python package  
src. The src package is a directory/folder that contains multiple MFS files/folders (a.k.a. modules)  
and can run independently of other packages. This allows us to host our application and its  
components: app (routes/endpoints), common (database), and models (charts and features) as  
modules within the src package. Thus, modularity has enhanced the functionality of MFS not only  
in terms of re-usability but also by making MFS a highly extensible program that enables users to  
add any customized function for any type of new feature.

Univariate analysis with core or non-core genes

A univariate approach evaluates a dataset by analyzing a single variable or column at a time. Several univariate methods such as Histograms, Count and Distribution plots, Box plots, and Violin plots were used to analyze the gene structural features. Four gene structure properties were chosen: the gene length, the exon number, the three-prime UTR length, and the five-prime UTR length from the “Gene Structure” drop-down menu, and “Pan-genome” as the label from the “Labels” drop-down menu, and then performed all univariate analyses sequentially from the “Choose Analysis” section. We conducted these analyses on both downsampled and full datasets, resulting in a grid of plots for each selected feature across different categories of the “Pan-genome” label (core genes, near-core genes, dispensable genes, and private genes) (Supplementary Figure S2). The plots summarize individual gene structures across the different categories of “Pan-genome” genes.

Bivariate analysis

We performed two types of bivariate analysis: Scatter plots and Joint plots on the gene structural features. Due to the nature of the bivariate analysis, we selected two gene structural features, the three-prime UTR length, and the five-prime UTR length, from the “Gene Structures” drop-down menu and the “Pan-genome” label from the “Labels” drop-down menu, and then performed each bivariate analysis individually from the “Choose Analysis” section. These analyses were conducted on both downsampled and full datasets. Our outputs were interactive plots illustrating the relationship between two selected variables, with each variable represented on its own axis. These plots are labeled according to the different categories of the “Pan-genome” label (core genes, near-core genes, dispensable genes, and private genes) to allow for simultaneous viewing of the pairs' relationships across the various “Pan-genome” genes categories (Supplementary Figure S3).

Multivariate analysis with core or non-core genes

As the name implies, multivariate analysis is used to visualize the relationship between more than two data variables. Using the “Gene Structures” drop-down menu, we selected five gene structures: gene length, exon number, three-prime UTR length, five-prime UTR length, CDS length, and the “Pan-genome” label from the “Labels” drop-down menu. We ran the multivariate analysis (Pair plots and Correlation plots) one by one from the “Choose Analysis” section. We conducted these analyses on both downsampled and full datasets. The multivariate plots provide a quick visual summary of the potential association between all selected features of interest, and like the bivariate analysis, the plots are labeled according to the different categories of the “Pan-genome” label (core genes, near-core genes, dispensable genes, and private genes) to allow for simultaneous viewing of the relationship between the selected features across the various categories of the “Pan-genome” label (Supplementary Figure S4). By simultaneously exploring gene structure features, we can observe that several features are significantly correlated in both core and non-core genes. Therefore, the plots can initially demonstrate how the different genomic features can significantly contribute to our understanding of core or non-core genes and highlight the potential for gene structural features in core/non-core gene classification.

Independent multivariate analysis

Heatmaps are another efficient way of carrying out multivariate analysis. In our application, the Heatmaps are generated for the first 100 genes of the selected label’s downsampled dataset. Supplementary Figure S4E presents a heatmap that illustrates the relationship of the first 100 “Pan-genome” genes with the selected gene structures: gene length, exon number, three-prime UTR length, and CDS length. The heatmap only displays the first 100 genes of the downsampled dataset due to time and data complexity constraints. However, users are free to customize the script to display more genes.

Exploratory analysis with unsupervised machine learning

Due to time and data complexity, the unsupervised machine-learning modules are only available for downsampled datasets of the selected label. In addition, the selected labels act as features during the application of unsupervised machine-learning modules. For example, upon selecting any label (“Pan-genome” in this case), the unsupervised ML modules are applied to the downsampled “Pan-genome” genes and will assist users in further identifying the distinct clusters within the “Pan-genome” genes according to the similarity among them. This module covers two major classes of techniques: clustering and dimension reduction. The clustering of numerical features was demonstrated with gene expression data using the Downsampled Dendrogram plots, Downsampled Hierarchical Scatter plots, and Downsampled Hierarchical Heatmaps (Supplementary Figure S5). Users can select their preferred gene expression data set. We will present an example of clustering categorical features using the Protein localization dataset. Gene clustering works as if the genes are not already clustered or grouped.

To perform clustering analysis with the numerical features, we selected the Fowler Lab dataset containing transcriptomics data from four male reproductive stages of maize (tassel primordia, microspores, mature pollen, and sperm cells) (1) from the “Gene Expression” dropdown menu. We ran all clustering algorithms: Downsampled Dendrogram plot, Downsampled Hierarchical Scatter plot, and Downsampled Hierarchical Heatmap from the “Choose Analysis” dropdown menu (Supplementary Figure S5). For the Downsampled Hierarchical Scatter plot, we need to specify the number of clusters we want to view. Moreover, this number of clusters corresponds to the number of clusters formed in the Dendrogram plot, so we perform this analysis only after studying the Dendrogram plot. The Downsampled Hierarchical Scatter plot is dynamically sized for the number of clusters, so every user can analyze the dataset based on how many clusters they believe the selected data is forming.

Based on the Dendrogram in Supplementary Figure S5A, we can see strong groups of genes forming across the dataset. Using the Dendrogram, we can visually determine how many groups we should focus on based on the height of the Dendrogram, where the heights reflect the distance between the clusters, as shown in the zoomed Dendrogram Supplementary Figure S5A. In general, the higher the height, the farther apart two objects or genes appear. For example, we can imagine sliding a horizontal line down the Dendrogram and cutting it into different groups. For simplicity, we can imagine three main clusters forming in our Dendrogram Supplementary Figure S5A, showing a big difference between the green cluster, red cluster, and blue cluster. Therefore, we use the same number of clusters as an input for our Downsampled Hierarchical Scatter plot (Supplementary Figure S5B).

The Hierarchical Scatter Plot (Supplementary Figure S5B) created with the Fowler Lab expression data is a Pair plot that visualizes the associations between different pairs of the lab tissues across the gene clusters formed during Dendrogram analysis. For example, the expression levels of genes in Cluster 0 are lower. Furthermore, Cluster 2 genes are expressed more in the Microspore than in the Sperm cell tissue.

The output from the Hierarchical Heatmap (Supplementary Figure S5C) allows us to simultaneously visualize clusters of samples (selected experiment tissues) in the column and features (genes) in the rows. Although the gene expression across the four tissues shows similar patterns in the Fowler data, upon closer inspection, we will be able to see some groups of genes upregulated in Tassel Primordia tissue and downregulated in Sperm Cell tissue. In addition, this plot allows us to see how tissues group together. The Tassel Primordia, Microspore, and Bicellular Male Gametophyte cluster together, whereas the Sperm cell tissue forms its own cluster. It can be important to see, for example, if all the replicates of an experiment cluster together.

Clustering analysis with categorical features was carried out using Protein localization features selected from the dropdown menu "Protein Localization" and "K-mode" clustering algorithms chosen from the dropdown menu "Choose Analysis.” The output (Supplementary Figure S5D) consists of k-mode bar plots displaying three gene clusters derived from the categorical features (subcel1, subcel2, subcel3, subcel4, subcel5, “Pan-genome”). The localization features subcel1, subcel2, subcel3, subcel4, subcel5, are comprised of predicted subcellular localizations such as nucleus, cytoplasm, extracellular, mitochondria, cell membrane, endoplasmic reticulum, plastid, golgi apparatus, lysosome/vacuole, peroxisome generated from the protein sequences using the tool WolfPsort (2). The X-axis on the K-mode bar plot represents the K-clusters within each category of selected Protein localization. The Y-axis displays the frequency of K-clusters within each category of Protein localization.

We selected the Fowler lab data set from the "Gene Expression" drop-down menu to conduct a principal component analysis and ran all available PCA analyses: PCA 2D variables cluster, PCA 2D observation cluster, PCA 2D biplot cluster, PCA 3D variables cluster, and PCA 3D observation cluster. Additionally, label selection is essential for PCA analysis to determine the relationship between several gene expression features aggregated into principal components on the various categories of core or non-core genes.

The PCA 2D variables cluster plot (Supplementary Figure S6A) shows the relationship or similarity between the variables (treatment or tissues), where the proximity of the points indicates similarity. The first two principal components account for 95% of the variation among treatments or tissues in the plot. In the plot, we can see that the Microspore, Bicellular male gametophyte and the Tassel Primordia tissues are positioned together, while the Sperm cells are positioned slightly farther along the first and the second principal component, respectively. Therefore, with PCA 2D variables cluster plots, we can visualize how certain tissues from the same lab cluster together.

PCA 2D observation cluster plot (Supplementary Figure S6B) depicts the correlation (or similarity) between observations (gene models) across multiple treatments. According to the plot, the first two PCs account for 86% of the total variation in the gene expression dataset, which is an excellent approximation of the variation present in the original 4D dataset. In this graph, a large majority of the data points are near each other, which indicates that the observations (gene expression values of the different “Pan-genome” genes) are similar (usually lower) across the four tissues. However, the core and near-core genes tend to have slightly higher expression levels across the four tissues than the private and the dispensable genes. Observations further out are either outliers or naturally extreme observations. Plot observations are annotated with shapes and colors to highlight gene models and gene types (core/non-core). In this way, we can easily hover over any data point, especially outliers, to find gene models with extreme gene expression values across various tissue types.

PCA 2D biplot (Supplementary Figure S6C-S6D) combines the PCA 2D variables cluster plots and the PCA 2D observation cluster plots. Lined vectors in the biplot represent the PCA 2D variables cluster plots. The biplot shows the relationship between all genes and treatments (core/non-core). Supplementary Figure S6C shows how the first principal component (x-axis) correlates to the mean response of genes (colors), accounting for 66.3% of the variation. The x-axis values for all treatments (vectors) are similar with positive values, and the data points are centered under the labels. Genes to the right of Supplementary Figure S6C are highly expressed, while genes to the left of the center are downregulated in all treatments, as determined by the gene score and treatment vector. In summary, the biplot displays the same information as the other two PCA plots but retains a greater amount of information about individual gene responses while also presenting global expression effects. The second axis, which accounts for an extra 20.1% of overall variation, has a lower impact on the gene responses contrast across the different treatments. The angles between the vectors or variables tell us how variables are related. When two vectors form a small angle, they represent highly correlated variables. They are unlikely to be correlated if they meet at 90°. They are negatively correlated when they diverge and form a large angle (close to 180°). The length of vectors in biplot refers to the amount of variance contributed by the vector. Generally, the more length the vector has, the more variance it contributes to space and represents it better.

Additionally, the PCA 3D variables cluster plot (Supplementary Figure S6E) and the PCA 3D observation cluster plot (Supplementary Figure S6F) are similar to the PCA 2D variables cluster plot and the PCA 2D observation cluster plot; however, instead of two principal components, now we can observe three principal components that contribute to most of the variance present in the original high-dimensional dataset.

References

1. Warman, C., Panda, K., Vejlupkova, Z., Hokin, S., Unger-Wallace, E., Cole, R.A., Chettoor, A.M., Jiang, D., Vollbrecht, E., Evans, M.M.S. *et al.* (2020) High expression in maize pollen correlates with genetic contributions to pollen fitness as well as with coordinated transcription from neighboring transposable elements. *PLoS Genet*, **16**, e1008462.

2. Horton, P., Park, K.J., Obayashi, T., Fujita, N., Harada, H., Adams-Collier, C.J. and Nakai, K. (2007) WoLF PSORT: protein localization predictor. *Nucleic Acids Res*, **35**, W585-587.