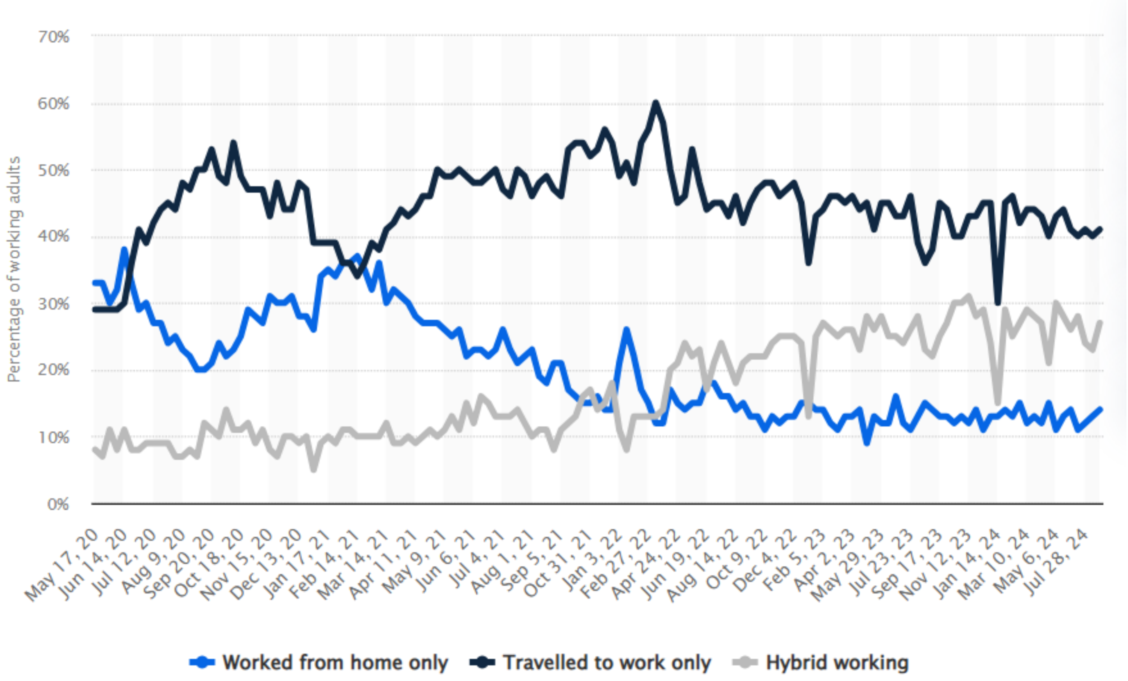
PREDICT HOMEWORKING: DESIGNING A FORECASTING MODEL

Aditi Shilke and Neha Jain

TABLE OF CONTENTS

INTRODUCTION

- There is a growing trend of hybrid work as shown by Statista's 2024 report, published by D. Clark, tracking UK remote, hybrid, and on-site trends (2020-2024).
- Most likely because there are significant benefits of remote work and on-site work as shown in Working from Home Around the Globe (2023) by Aksoy et al.
- Thus, effectively adapting the resource planning practices to Hybrid working has become paramount
- But the same study also shows that average number of WFH days per week that employees desire skews higher than Average number of WFH days per week that employers plan.
- Such a misalignment can lead to Reduced Job Satisfaction and Morale, Higher Turnover Rates and Challenges in Attracting Talent. Thus, a forecasting model that can predict whether an employee will work from home on the next day can help align to employee’s preference while optimizing resource planning.



CLASSIFICATION PROBLEM

REASONING

Binary Outcome
(Discrete Labels)

i.e. WFH or not WFH

Pattern-Based Decision

Using historical data (such as tenure, salary, WFH in the past) to predict the outcome.

CLASSIFICATION PROBLEM

SOLUTIONS

- **k-Nearest Neighbours**
- **C5.0**
- **CART**
- **Random Forest**
- **Logistic Regression**
- **Neural Network**
- **Support Vector Machine**

VARIABLE DEFINITIONS

PREDICTOR VARIABLES

distance_office: Quantitative feature, distance in kilometers from the employee's residence to the workplace

salary_range: Categorical variable, the employee's yearly income range in euros

gas_price: Continuous feature, the price of gas per liter near the employee's residence

public_transportation_cost: Continuous variable, cost of public transport in euros

wfh_prev_workday: Binary feature, indicates whether the employee worked from home on the previous workday

workday: Categorical variable, the day of the week for which the prediction is being made

tenure: Quantitative feature, the number of years the employee has been with the company

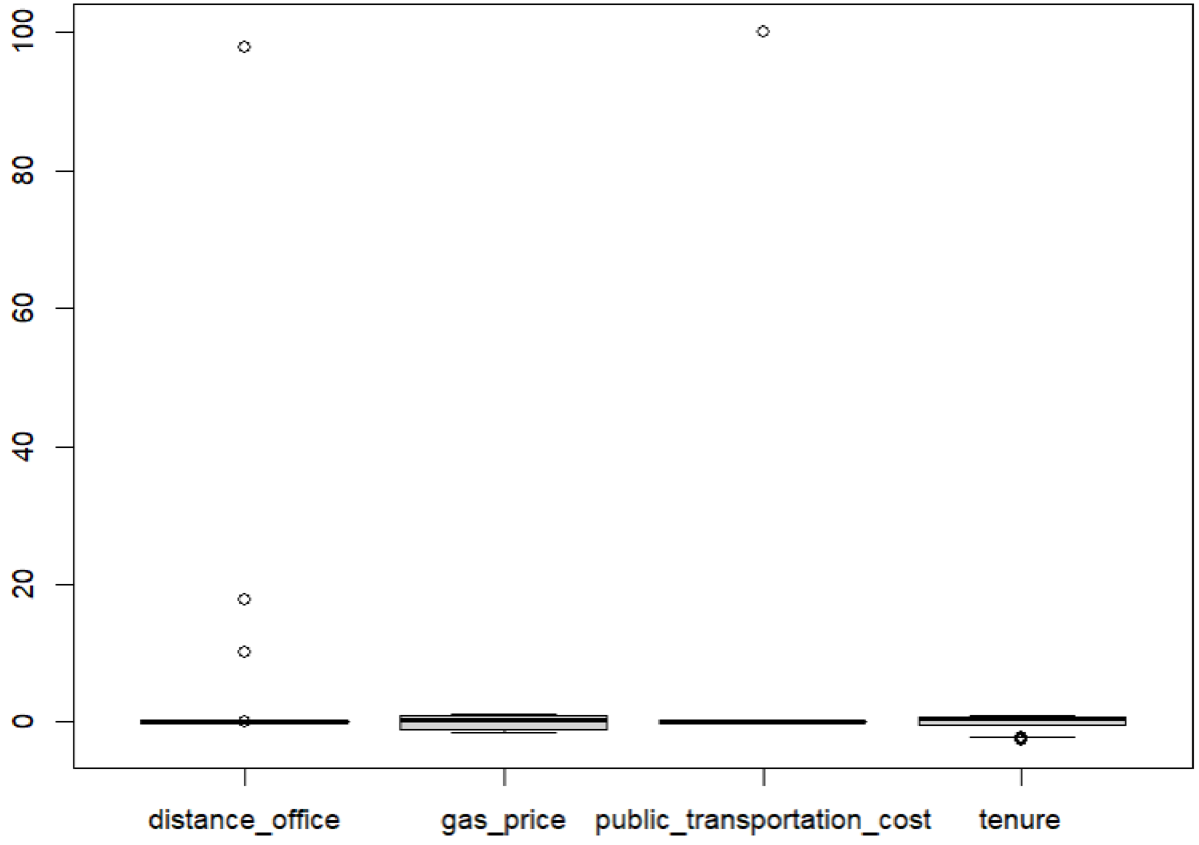
TARGET VARIABLE

work_home:
The binary outcome variable that the model aims to predict, indicating whether the employee worked from home (1) or not (0) on a specific day.

DATA SUMMARY

```
> summary(data_ori)
   identifier     distance_office      salary_range           gas_price    
 Min.   :    1   Min.   :      0.0   Length:10000       Min.   :1.400  
 1st Qu.: 2501   1st Qu.:      0.9   Class :character   1st Qu.:1.595  
 Median : 5000   Median :      2.4   Mode  :character   Median :2.071  
 Mean   : 5000   Mean   :     53.9                      Mean   :1.975  
 3rd Qu.: 7500   3rd Qu.:      5.6                      3rd Qu.:2.324  
 Max.   :10000   Max.   :389563.8                       Max.   :2.400  
                 NA's   :1                                             
 public_transportation_cost wfh_prev_workday     workday               tenure        
 Min.   :      4            Length:10000       Length:10000       Min.   :0.005116  
 1st Qu.:      5            Class :character   Class :character   1st Qu.:4.332927  
 Median :      8            Mode  :character   Mode  :character   Median :5.923641  
 Mean   :    422                                                  Mean   :5.126710  
 3rd Qu.:      9                                                  3rd Qu.:6.601942  
 Max.   :4152345                                                  Max.   :6.997065  
                                                                  NA's   :1         
 work_home  
 no :7500  
 yes:2500  
```

HANDLING OUTLIERS



Outlier Removal: In this case, outliers are extreme deviations from the feature distributions, which posed a risk of skewing model learning. Due to their minimal count, these are excluded to avoid biased model interpretations.

```
> summary(data)
   identifier     distance_office      salary_range          gas_price    
 Min.   :    1   Min.    : 0.003997   Length:9994        Min.   :1.400  
 1st Qu.: 2502   1st Qu.: 0.948561   Class :character   1st Qu.:1.595  
 Median : 5000   Median : 2.403124   Mode  :character   Median :2.071  
 Mean   : 5001   Mean    : 3.902467                      Mean   :1.975  
 3rd Qu.: 7499   3rd Qu.: 5.592904                      3rd Qu.:2.324  
 Max.   :10000   Max.    :19.671320                      Max.   :2.400  
 public_transportation_cost wfh_prev_workday     workday         
 Min.   :4.003              Length:9994        Length:9994       
 1st Qu.:5.015              Class :character   Class :character  
 Median :7.820              Mode  :character   Mode  :character  
 Mean   :6.977                                                   
 3rd Qu.:8.531                                                   
 Max.   :8.998                                                   
     tenure          work_home 
 Min.   :0.005116   no :7494  
 1st Qu.:4.332774   yes:2500  
 Median :5.924389             
 Mean   :5.126773             
 3rd Qu.:6.602244             
 Max.   :6.997065             
>
```

HANDLING MISSING VALUES

```
> data_ori[data_ori == ""] = NA
> # check for missing values
> colSums(is.na(data_ori))
              identifier          distance_office             salary_range
                       0                        1                        0
               gas_price public_transportation_cost          wfh_prev_workday
                       0                        0                        0
                 workday                   tenure                work_home
                       3                        1                        0
```

Missing Value Treatment: With only five missing entries, deletion was chosen over (mean, median, mode, knn, predictive) imputation. This approach preserved dataset consistency while minimizing imputation bias.

```
> # removing rows with missing values
> data = na.omit(data_ori)
> colSums(is.na(data))
              identifier          distance_office             salary_range
                       0                        0                        0
               gas_price public_transportation_cost          wfh_prev_workday
                       0                        0                        0
                 workday                   tenure                work_home
                       0                        0                        0
```

ENCODING

identifier	distance_office	salary_range	gas_price	public_transportation_cost	wfh_prev_workday	workday	tenure	work_home
1	5.646414716	40K-60K	2.380145	4.382271	True	Tuesday	5.40871770	no
2	10.313797270	0-20K	1.493386	8.763065	False	Thursday	5.82378149	no
3	2.738644304	20K-40K	2.115532	4.245992	True	Thursday	6.53259372	no
4	3.328993318	20K-40K	1.753234	4.692132	True	Friday	6.38781226	no
5	10.256253260	0-20K	2.223889	8.811094	True	Thursday	6.83348304	yes
6	2.019772177	20K-40K	2.364352	5.861525	False	Tuesday	4.93073965	no

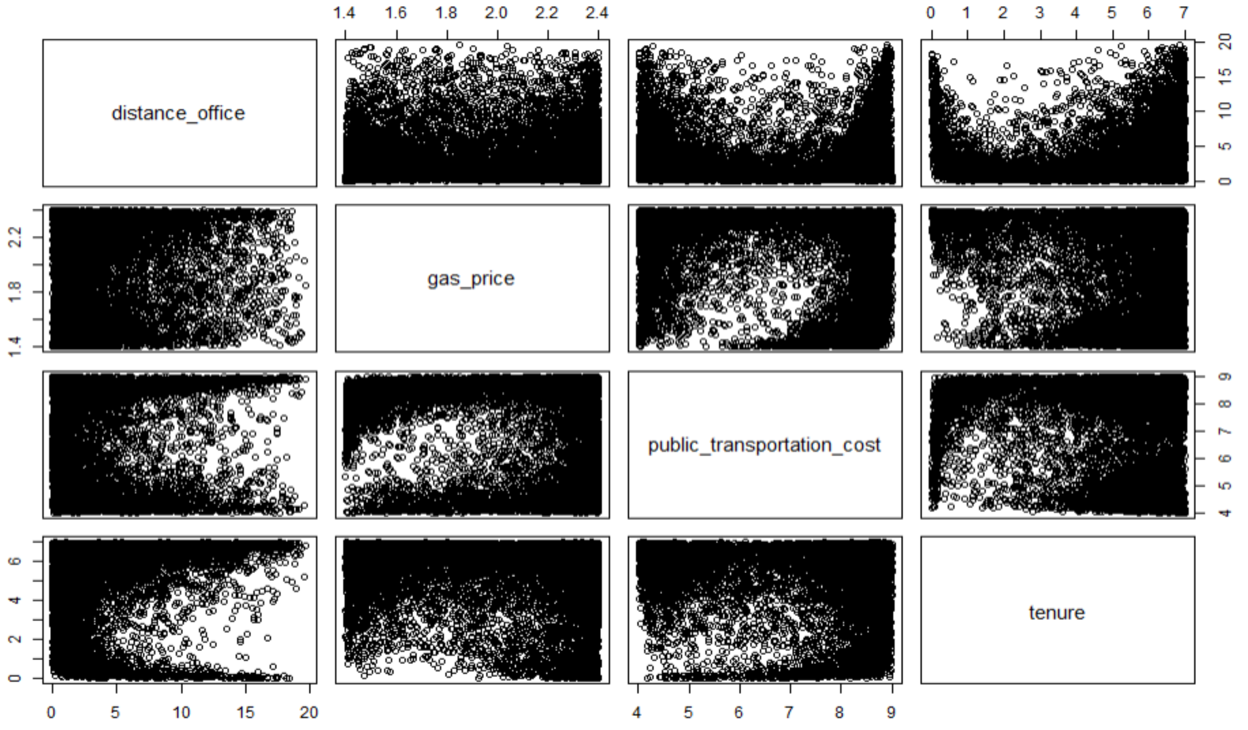
```
> summary(tranform_data)
   identifier     distance_office      salary_range20K.40K salary_range40K.60K
 Min.   :    1   Min.   : 0.003997   Min.   :0.0000      Min.   :0.0000
 1st Qu.: 2502   1st Qu.: 0.948446   1st Qu.:0.0000      1st Qu.:0.0000
 Median : 5000   Median : 2.403124   Median :0.0000      Median :0.0000
 Mean   : 5000   Mean   : 3.902763   Mean   :0.2596      Mean   :0.1685
 3rd Qu.: 7497   3rd Qu.: 5.594566   3rd Qu.:1.0000      3rd Qu.:0.0000
 Max.   :10000   Max.   :19.671320   Max.   :1.0000      Max.   :1.0000
 salary_range60K.    gas_price     public_transportation_cost wfh_prev_workdayTrue
 Min.   :0.00000   Min.   :1.400   Min.   :4.003               Min.   :0.0000
 1st Qu.:0.00000   1st Qu.:1.595   1st Qu.:5.015               1st Qu.:0.0000
 Median :0.00000   Median :2.070   Median :7.820               Median :1.0000
 Mean   :0.08487   Mean   :1.975   Mean   :6.976               Mean   :0.5561
 3rd Qu.:0.00000   3rd Qu.:2.324   3rd Qu.:8.531               3rd Qu.:1.0000
 Max.   :1.00000   Max.   :2.400   Max.   :8.998               Max.   :1.0000
 workdayMonday    workdayThursday  workdayTuesday   workdayWednesday
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.000
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000
 Median :0.0000   Median :0.0000   Median :0.0000   Median :0.000
 Mean   :0.2118   Mean   :0.1928   Mean   :0.1891   Mean   :0.194
 3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.000
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.000
     tenure
 Min.   :0.005116
 1st Qu.:4.333079
 Median :5.924389
 Mean   :5.126855
 3rd Qu.:6.602376
 Max.   :6.997065
```

Label Encoding: Label encoding assigns ordinal values to categories, which introduces an artificial rank. This could mislead the model as categorical features like *salary_range* lack any inherent ranking.

One-Hot Encoding: One-hot encoding is ideal here because it creates binary indicators for each category, which allows the model to interpret categorical variables without assuming any order or hierarchy.

With full rank encoding, for example, the binary variable for *Friday* is dropped to prevent perfect collinearity among weekday indicators as multicollinearity may make the model unstable.

MULTICOLINEARITY CHECK




```
> cor(data[,c('distance_office','gas_price','public_transportation_cost','tenure')])
                           distance_office    gas_price public_transportation_cost      tenure
distance_office                1.000000000  0.008907137                 0.01586766  0.08526859
gas_price                      0.008907137  1.000000000                -0.36680840 -0.20898335
public_transportation_cost     0.015867659 -0.366808396                 1.00000000 -0.15407154
tenure                         0.085268588 -0.208983351                -0.15407154  1.00000000
```

Scatterplot matrix: The scatterplots show no clear linear relationships among the variables, indicating limited correlations. The *distance from office* shows a scattered pattern with *gas price*, while it has a denser pattern with *public transportation cost*, suggesting limited influence from *distance* on costs. Similarly, the scatterplots of *gas price* versus *public transportation cost* and *public transportation cost* versus *tenure* indicate mild clustering but no definitive trends.

Correlation Matrix: The correlation coefficients reveal no significant correlations among the variables, with all correlation coefficients being relatively low. The highest correlation is between *gas_price* and *public_transportation_cost*, indicating a weak inverse relationship, suggesting that as gas prices increase, public transportation costs might decrease slightly. Overall, these findings imply that multicollinearity is not a concern, allowing for more stable model training without redundancy among predictors.

SAMPLING METHOD

Systematic Sampling: May miss patterns in the minority class (*work_home* = "yes") due to regular intervals, leading to underrepresentation.

Simple Random Sampling: Unlikely to address the imbalance effectively and may result in a minority class that's too small for reliable training.

Proportional Sampling: Preserves the original imbalance, which fails to give the minority class equal weight in training.

Upsampling: Risks overfitting by duplicating minority class data, which can reduce the model's generalizability.

Downsampling: Preferable because the target variable (*work_home*) is imbalanced, with significantly more "no" than "yes" entries, and downsampling ensures a balanced dataset while avoiding data duplication or overrepresentation

```
> summary(data$work_home)
  no  yes
7492 2500
> dim(data)
[1] 9992   13
```

```
> summary(data_down$work_home)
  no  yes
2500 2500
> dim(data_down)
[1] 5000   13
```

SCALING

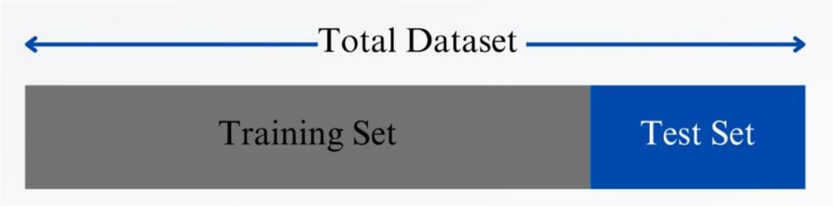
Scaling is essential for this dataset to ensure that features with varying magnitudes do not disproportionately influence distance calculations and model performance. By rescaling the features to a uniform range, we enhance the accuracy and convergence speed of algorithms sensitive to feature scaling.

Min-Max Normalization vs. Standardization:

Min-Max normalization is ideal here to rescale the features into a [0, 1] range, which is particularly useful for algorithms like neural networks that assume input values are within a specific range. Unlike standardization, which transforms data to have a mean of 0 and standard deviation of 1, Min-Max normalization preserves the original relationships among the features while ensuring all values are positively bounded.

```
> # normalizing the data
> preProcValues <- preProcess(data_down, method = "range")
> data_down <- predict(preProcValues, data_down)
> summary(data_down)
 distance_office    salary_range20K.40K salary_range40K.60K salary_range60K.   gas_price
 Min.    :0.00000   Min.    :0.0000     Min.    :0.000      Min.    :0.0000   Min.    :0.0000
 1st Qu.:0.04591   1st Qu.:0.0000      1st Qu.:0.000       1st Qu.:0.0000    1st Qu.:0.3665
 Median :0.12338   Median :0.0000      Median :0.000       Median :0.0000    Median :0.7912
 Mean   :0.20247   Mean   :0.3016      Mean   :0.195       Mean   :0.1028    Mean   :0.6485
 3rd Qu.:0.29083   3rd Qu.:1.0000      3rd Qu.:0.000       3rd Qu.:0.0000    3rd Qu.:0.9397
 Max.   :1.00000   Max.   :1.0000      Max.   :1.000       Max.   :1.0000    Max.   :1.0000
 public_transportation_cost wfh_prev_workdayTrue workdayMonday    workdayThursday
 Min.   :0.0000             Min.   :0.0000       Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.4181             1st Qu.:0.0000       1st Qu.:0.0000   1st Qu.:0.0000
 Median :0.8189             Median :1.0000       Median :0.0000   Median :0.0000
 Mean   :0.6660             Mean   :0.5532       Mean   :0.2066   Mean   :0.1982
 3rd Qu.:0.9257             3rd Qu.:1.0000       3rd Qu.:0.0000   3rd Qu.:0.0000
 Max.   :1.0000             Max.   :1.0000       Max.   :1.0000   Max.   :1.0000
 workdayTuesday   workdayWednesday    tenure          work_home
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   no :2500
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.4072   yes:2500
 Median :0.0000   Median :0.0000   Median :0.7965
 Mean   :0.1904   Mean   :0.1988   Mean   :0.6588
 3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.9322
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

PARTITIONING THE DATA




```
> summary(training$work_home)
  no  yes
1875 1875
> dim(training)
[1] 3750   13
> summary(test$work_home)
 no yes
625 625
> dim(test)
[1] 1250   13
```

Splitting the dataset into training and testing sets allows us to **assess the model's ability to generalize**. By training on one subset and testing on another, we can check if the model performs well on new data, **reducing the risk of overfitting.**

A **75-25 split** allocates more data for training, helping the model learn more complex patterns. The remaining 25% is sufficient for reliable evaluation, balancing the need for model accuracy with a robust assessment of generalizability.

HYPERPARAMETER TUNING

Accuracy for the Test Dataset:

	repeated cv	grid search	adaptive cv	random search
KNN	0.904	0.904	0.904	0.904
C5.0	0.92	0.92	0.92	0.92
CART	0.8456	0.9024	0.8456	0.9024
Random Forest	0.916	0.9176	0.9168	0.9176
Neural Network	0.9128	0.9256	0.9128	0.9256
SVM (Linear)	0.876	0.876	0.8776	0.8776
SVM (Radial)	0.9152	0.9152	0.9152	0.9152

- **Repeated Cross Validation:** This method is effective for improving reliability but can be computationally expensive and time-consuming. For large datasets, it may slow down the hyperparameter tuning process unnecessarily.

- **Grid Search:** Grid search is ideal for hyperparameter tuning as it exhaustively explores all combinations, ensuring that the best parameters are identified. This comprehensive approach often leads to better model performance compared to other methods.

- **Adaptive Cross Validation:** While adaptive cross-validation can adjust sampling strategies, it may introduce complexity without significantly enhancing model performance. The additional computational cost may not justify the marginal gains in accuracy.

- **Random Search:** Random search can be efficient in exploring parameter spaces but may miss the optimal combinations due to its stochastic nature. Unlike grid search, it does not guarantee finding the best parameters, making it less reliable for precise tuning.

K-NEAREST NEIGHBOURS

```
k-Nearest Neighbors

3750 samples
  12 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3374, 3374, 3375, 3375, 3374, 3375, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   3  0.9197309  0.8394630
   5  0.9229288  0.8458565
   7  0.9167911  0.8335826
   9  0.9167940  0.8335884
  11  0.9146628  0.8293256
  13  0.9138649  0.8277283
  15  0.9135997  0.8271987
  17  0.9103989  0.8207965
  19  0.9087982  0.8175950
  21  0.9074670  0.8149333
  23  0.9055939  0.8111856
  25  0.9039975  0.8079920

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 5.
```

Why KNN?

- It's a straightforward classification algorithm that can handle both numerical and binary categorical features without needing complex transformations.
- In predicting a binary target (*work_home*), KNN can use proximity in feature space (e.g., *distance_office and gas_price*) to make predictions, which may naturally capture patterns based on these spatial and economic factors.

Hyperparameter tuning:

- The choice of k directly impacts the model's bias-variance tradeoff: smaller values of k may lead to overfitting, while larger values may underfit.
- The caret package tunes k by performing a grid search across specified k values, using cross-validation to evaluate performance for each k and selecting the one with the highest accuracy.

K-NEAREST NEIGHBOURS

```
> acctr <- confusionMatrix(prediction.train, training[,13])
> acctr$table
          Reference
Prediction   no  yes
       no  1764  117
       yes  111 1758
> acctr$overall['Accuracy']
Accuracy
  0.9392
> accte <- confusionMatrix(prediction.test, test[,13])
> accte$table
          Reference
Prediction  no yes
       no  568  63
       yes  57 562
> accte$overall['Accuracy']
Accuracy
   0.904
```

- The confusion matrix for the training set reveals that the KNN model correctly classified 1764 "no" and 1758 "yes" cases, with a few misclassifications (117 false positives and 111 false negatives), resulting in a strong accuracy of 93.92%.

- For the test set, the model achieved an accuracy of 90.4%, with 568 correct "no" and 562 correct "yes" predictions, and a relatively low misclassification rate (63 false positives and 57 false negatives).

- This consistency in high training and test accuracy suggests good model generalization and effective tuning of the k value, minimizing overfitting while capturing meaningful patterns.

C5.0

```
C5.0

3750 samples
  12 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3374, 3374, 3375, 3375, 3374, 3375, ...
Resampling results across tuning parameters:

  model  trials  Accuracy   Kappa
  rules  1       0.9218571  0.8437162
  rules  2       0.9183947  0.8367916
  rules  3       0.9191947  0.8383883
  rules  4       0.9218542  0.8437098
  rules  5       0.9296011  0.8592015
  tree   1       0.9146599  0.8293224
  tree   2       0.9191975  0.8383977
  tree   3       0.9191933  0.8383903
  tree   4       0.9223961  0.8447920
  tree   5       0.9234635  0.8469269

Tuning parameter 'winnow' was held constant at a value of FALSE
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were trials = 5, model = rules and winnow = FALSE.
```

Why C5.0?

- Highly interpretable decision rules, which can aid in understanding the relationships between predictors and the target variable, allowing for straightforward insights into decision-making.
- It automatically handles feature selection, identifying the most relevant variables while building the model, which can simplify the modeling process and enhance performance without extensive manual feature engineering.

Hyperparameter tuning:

- model influences the structure and interpretability of the decision rules, while trials controls the number of trees, enhancing the model's ability to capture complex patterns and reducing overfitting.
- The caret package uses a grid search approach to evaluate combinations of these hyperparameters and has identified model=rules and trials=5 as the optimal configuration to maximize predictive performance on the validation dataset.

C5.0

```
> acctr <- confusionMatrix(prediction.train, training[,13])
> acctr$table
          Reference
Prediction   no  yes
       no  1792   76
       yes   83 1799
> acctr$overall['Accuracy']
Accuracy
  0.9576
> accte <- confusionMatrix(prediction.test, test[,13])
> accte$table
          Reference
Prediction  no yes
       no  572  47
       yes  53 578
> accte$overall['Accuracy']
Accuracy
    0.92
```

```
> acctr <- confusionMatrix(prediction.train, training[,13])
> acctr$table
          Reference
Prediction   no  yes
       no  1794   68
       yes   81 1807
> acctr$overall['Accuracy']
 Accuracy
0.9602667
> accte <- confusionMatrix(prediction.test, test[,13])
> accte$table
          Reference
Prediction  no yes
       no  569  50
       yes  56 575
> accte$overall['Accuracy']
Accuracy
  0.9152
```

When using **winnow = FALSE**, the model achieved an impressive training accuracy of 95.76% and a test accuracy of **92.00%**. In contrast, with **winnow = TRUE**, the training accuracy slightly improved to 96.03%, but the test accuracy decreased to **91.52%**. Thus, while winnowing can enhance training performance, it may lead to a slight drop in generalization ability on unseen data. We opted to keep **winnow = FALSE** as it provided robustness and reliability in predicting the work-from-home outcomes across both training and test datasets.

```
> report
          Model Acc.Train Acc.Test
Accuracy   k-NN    0.9392    0.904
Accuracy1  C5.0    0.9576    0.920
```

CART

```
CART

3750 samples
  12 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3374, 3374, 3375, 3375, 3374, 3375, ...
Resampling results across tuning parameters:

  cp     Accuracy   Kappa
  0.001  0.9093259  0.8186477
  0.010  0.8816052  0.7632035
  0.050  0.8525299  0.7050542
  0.100  0.8525299  0.7050542

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.001.
```

```
> acctr <- confusionMatrix(prediction.train, training[,13])
> acctr$table
          Reference
Prediction   no  yes
       no  1773  131
       yes  102 1744
> acctr$overall['Accuracy']
 Accuracy
0.9378667
> accte <- confusionMatrix(prediction.test, test[,13])
> accte$table
          Reference
Prediction  no yes
       no  565  62
       yes  60 563
> accte$overall['Accuracy']
Accuracy
  0.9024
```

Why CART?

Especially suitable for this dataset because it builds a single, interpretable decision tree, which can be less complex than C5.0’s boosted tree ensemble. This simplicity aids interpretability.

Hyperparameter Tuning:

The CART model’s complexity parameter (cp) was tuned to control the pruning of the decision tree, helping to avoid overfitting by pruning less relevant branches, which was optimized in caret for the best accuracy.

Results:

- With an optimal cp of 0.001, CART achieved 93.79% training accuracy and 90.24% test accuracy, providing a balanced generalization.
- Additionally, the pruned tree structure avoids unnecessary complexity, highlighting only the most impactful features, which makes it suitable for deployment and easier for stakeholders to interpret.

RANDOM FOREST

```
maxnodes= 8    Accuracy= 0.8845222    mtry= 2

maxnodes= 16    Accuracy= 0.8989222    mtry= 2

maxnodes= 24    Accuracy= 0.9074585    mtry= 2

maxnodes= 100    Accuracy= 0.9263954    mtry= 10
```

```
Random Forest

3750 samples
  12 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3374, 3374, 3375, 3375, 3374, 3375, ...
Resampling results:

  Accuracy   Kappa
  0.9263961  0.8527902
```

Why Random Forest?

- Random Forest is suitable for this dataset as it can effectively handle complex interactions between predictors and is robust against overfitting, making it ideal for predicting binary outcomes like work-from-home decisions.
- Additionally, its ability to provide variable importance metrics can help identify the most influential factors impacting the target variable.

Hyperparameters tuning:

- The primary hyperparameters for the Random Forest model are `mtry`, which specifies the number of predictors to randomly sample for each tree, and `maxnodes`, which limits the maximum number of terminal nodes in the trees to prevent excessive growth and complexity.
- Hyperparameter tuning was conducted through a grid search approach. This allowed for identifying the optimal combination that maximizes model accuracy while ensuring effective generalization on unseen data.

RANDOM FOREST

```
> acctr <- confusionMatrix(prediction.train, training[,13])
> acctr$table
          Reference
Prediction   no  yes
       no  1768   63
       yes  107 1812
> acctr$overall['Accuracy']
 Accuracy
0.9546667
> accte <- confusionMatrix(prediction.test, test[,13])
> accte$table
          Reference
Prediction  no yes
       no  562  40
       yes  63 585
> accte$overall['Accuracy']
Accuracy
  0.9176
```

- The Random Forest model achieves a training accuracy of approximately 95.47%, indicating a strong fit to the training data with 1,812 true positives and 1,768 true negatives. In the test dataset, the model maintained a solid accuracy of around 91.76%, with 585 true positives and 562 true negatives, showcasing its ability to generalize well to unseen data.

- The low number of false positives (107) and false negatives (63) in the training set further emphasizes that the model effectively distinguishes between the two classes, which adds to its reliability for predicting work-from-home outcomes in real-world applications.

LOGISTIC REGRESSION

```
Generalized Linear Model

3750 samples
  12 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3374, 3374, 3375, 3375, 3374, 3375, ...
Resampling results:

  Accuracy   Kappa
  0.8815867  0.7631752
```

```
> acctr <- confusionMatrix(prediction.train, training[,13])
> acctr$table
          Reference
Prediction   no  yes
       no  1666  228
       yes  209 1647
> acctr$overall['Accuracy']
 Accuracy
0.8834667
> accte <- confusionMatrix(prediction.test, test[,13])
> accte$table
          Reference
Prediction  no yes
       no  548  76
       yes  77 549
> accte$overall['Accuracy']
Accuracy
  0.8776
```

Why Logistic regression?

- It is well-suited for the dataset because the target variable, work_home, is binary, with two classes: "yes" and "no." Unlike more complex models, logistic regression provides interpretable coefficients, which allows to understand the influence of each predictor on the probability of working from home.
- Additionally, it requires fewer computational resources, making it a practical choice for initial analysis and baseline performance.

Results:

The logistic regression model achieved an accuracy of 88.35% on the training set and 87.76% on the test set, indicating good generalizability. The confusion matrix shows that the model accurately predicts most cases, with only a small number of misclassifications, suggesting it performs reliably on this data.

NEURAL NETWORK

```
Neural Network

3750 samples
  12 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3374, 3374, 3375, 3375, 3374, 3375, ...
Resampling results across tuning parameters:

  size  decay  Accuracy   Kappa
  1     0.000  0.8446855  0.6893704
  1     0.001  0.8823860  0.7647714
  1     0.010  0.8821193  0.7642378
  1     0.100  0.8815853  0.7631699
  3     0.000  0.9087939  0.8175866
  3     0.001  0.9085137  0.8170275
  3     0.010  0.9117287  0.8234570
  3     0.100  0.9130571  0.8261115
  5     0.000  0.9178450  0.8356899
  5     0.001  0.9205224  0.8410433
  5     0.010  0.9221152  0.8442311
  5     0.100  0.9221245  0.8442489
  7     0.000  0.9213316  0.8426620
  7     0.001  0.9242642  0.8485302
  7     0.010  0.9317231  0.8634480
  7     0.100  0.9298635  0.8597275
  9     0.000  0.9293288  0.8586592
  9     0.001  0.9234614  0.8469218
  9     0.010  0.9282664  0.8565309
  9     0.100  0.9311969  0.8623937

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were size = 7 and decay = 0.01.
```

Why Neural Networks?

- Neural networks are suitable for the dataset due to their ability to model complex, non-linear relationships among features, which is particularly useful when handling multi-dimensional data.
- Unlike traditional classifiers, neural networks can automatically learn feature interactions, making them ideal for datasets with diverse predictor variables and underlying patterns, as seen in this dataset.

Hyperparameter Tuning:

- The hyperparameter tuning results indicate that a neural network with a size of 7 neurons and a decay rate of $1e{-}02$ yields the highest accuracy (approximately 93.17%).
- This suggests that a moderately complex architecture effectively balances bias and variance, capturing essential patterns in the data while minimizing overfitting.

NEURAL NETWORK

```
> acctr <- confusionMatrix(prediction.train, training[,13])
> acctr$table
          Reference
Prediction   no  yes
       no  1757  102
       yes  118 1773
> acctr$overall['Accuracy']
 Accuracy
0.9413333
> accte <- confusionMatrix(prediction.test, test[,13])
> accte$table
          Reference
Prediction  no yes
       no  579  47
       yes  46 578
> accte$overall['Accuracy']
Accuracy
  0.9256
```

- The model output indicates a training accuracy of 94.13% and a test accuracy of 92.56%, demonstrating strong predictive performance and generalization capabilities.
- The confusion matrices show that the model correctly classifies a high proportion of instances in both training and test datasets, with relatively few false positives and false negatives.
- Specifically, the training confusion matrix shows that the model has a high true positive rate (1757 out of 1875 "yes" cases), while the test confusion matrix indicates similar performance with 579 correct "yes" predictions.
- This suggests that while the model excels at classifying the majority class, it maintains a satisfactory level of performance for the minority class as well highlighting its well-rounded performance.

SUPPORT VECTOR MACHINE

```
Support Vector Machines with Linear Kernel

3750 samples
  12 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3374, 3374, 3375, 3375, 3374, 3375, ...
Resampling results across tuning parameters:

  C      Accuracy   Kappa
   0.01  0.8847853  0.7695731
   0.05  0.8847860  0.7695711
   0.10  0.8839860  0.7679724
   0.25  0.8834520  0.7669044
   0.50  0.8847853  0.7695700
   1.00  0.8847860  0.7695711
   1.50  0.8850527  0.7701049
   2.00  0.8850527  0.7701049
   5.00  0.8850527  0.7701049
  10.00  0.8847860  0.7695718

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was C = 1.5.
```

```
Support Vector Machines with Radial Basis Function Kernel

3750 samples
  12 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3374, 3374, 3375, 3375, 3374, 3375, ...
Resampling results across tuning parameters:

  sigma  C       Accuracy   Kappa
  0.05     2.00  0.9255940  0.8511876
  0.05     5.00  0.9306643  0.8613293
  0.05    10.00  0.9319948  0.8639910
  0.05   100.00  0.9335976  0.8671961
  0.10     0.01  0.8807902  0.7615811
  0.10     0.05  0.9119939  0.8239875

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.05 and C = 100.
```

Why SVM?

The presence of both linear and non-linear relationships among features can benefit from SVM's ability to use different kernels, allowing it to capture complex boundaries between classes. It's also less prone to overfitting in high-dimensional datasets, compared to algorithms like k-NN.

Kernel Selection:

- The linear SVM achieved a maximum accuracy of approximately 88.5% with a chosen cost parameter C of 1.5, indicating that it effectively separates the two classes ('no' and 'yes') in our dataset. The radial SVM model demonstrated significant improvement over lower parameter values, achieving a maximum accuracy of about 93.4% with a sigma of 0.05 and C of 100.
- The results indicate that the radial basis function (RBF) kernel outperforms the linear kernel, achieving a maximum accuracy of 93.4% compared to 88.5% for the linear kernel. This suggests that the RBF kernel is more effective in capturing complex patterns in the data, resulting in better classification performance.

SUPPORT VECTOR MACHINE

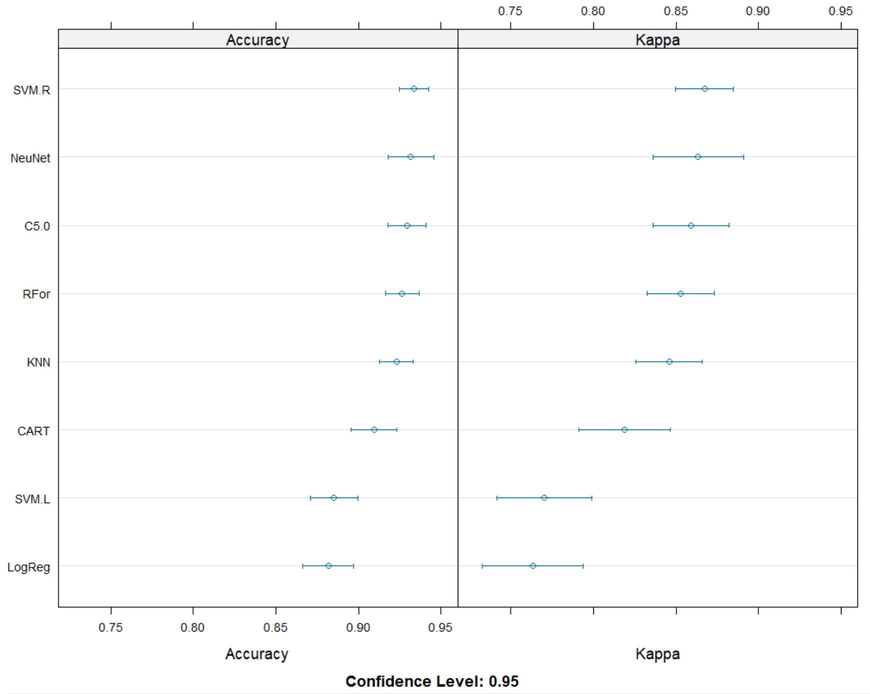
```
> acctr <- confusionMatrix(prediction.train, training[,13])
> acctr$table
          Reference
Prediction   no  yes
       no  1663  223
       yes  212 1652
> acctr$overall['Accuracy']
Accuracy
   0.884
> accte <- confusionMatrix(prediction.test, test[,13])
> accte$table
          Reference
Prediction  no yes
       no  546  76
       yes  79 549
> accte$overall['Accuracy']
Accuracy
   0.876
```

```
> acctr <- confusionMatrix(prediction.train, training[,13])
> acctr$table
          Reference
Prediction   no  yes
       no  1769   69
       yes  106 1806
> acctr$overall['Accuracy']
 Accuracy
0.9533333
> accte <- confusionMatrix(prediction.test, test[,13])
> accte$table
          Reference
Prediction  no yes
       no  566  46
       yes  59 579
> accte$overall['Accuracy']
Accuracy
   0.916
```

- The linear SVM achieved an accuracy of 88.4% on the training set and 87.6% on the test set, indicating good performance but a relatively higher misclassification rate compared to the radial kernel, as seen in the confusion matrix where 223 'yes' predictions were incorrectly classified as 'no' on the training set.
- The radial SVM showed significantly improved accuracy, with 95.3% on the training set and 91.6% on the test set, demonstrating its ability to capture complex patterns in the data effectively, as reflected in the confusion matrix where only 69 'yes' predictions were incorrectly classified as 'no' on the training set.
- **The radial SVM consistently outperformed the linear SVM** in both training and test accuracies, highlighting its superiority in handling the underlying complexities of the dataset, with the confusion matrix illustrating fewer misclassifications compared to the linear model, particularly for the 'yes' class.

ACCURACY ANALYSIS

```
> report
                         Model Acc.Train Acc.Test
Accuracy                  k-NN 0.9392000   0.9040
Accuracy1                 C5.0 0.9576000   0.9200
Accuracy2                 CART 0.9378667   0.9024
Accuracy3        Random Forest 0.9546667   0.9176
Accuracy4  Logistic Regression 0.8834667   0.8776
Accuracy5       Neural Network 0.9413333   0.9256
Accuracy6         SVM (Linear) 0.8840000   0.8760
Accuracy7         SVM (Radial) 0.9533333   0.9160
```



Accuracy and Robustness: The Neural Network has the highest test accuracy (0.9256) compared to all other models, indicating strong predictive power for new data. C5.0 and Random Forest also perform well (0.9200 and 0.9176, respectively), but Neural Network's slight edge suggests it generalizes best on this dataset.

Complexity and Interpretability: Neural Networks and Random Forests are known for handling complex patterns well, particularly with mixed data types (numerical, categorical), as seen here. Although Neural Networks can be less interpretable than other methods, their high test accuracy for this data and problem type makes them suitable for predictive precision, especially if interpretability is less of a priority.

Consistency Across Train and Test: Neural Network and C5.0 have relatively consistent training and test accuracies, suggesting these models are not overfitting, unlike k-NN which has a large train-test accuracy gap.

CONFUSION MATRIX ANALYSIS

```
          Reference
Prediction  no yes
       no  579  47
       yes  46 578
```

Neural Network

```
          Reference
Prediction  no yes
       no  572  47
       yes  53 578
```

C5.0

```
          Reference
Prediction  no yes
       no  562  40
       yes  63 585
```

Random Forest

Focusing on the models with higher accuracy and analysing their confusion matrix can also provide further insights into performance assessment.

C5.0: Correctly predicts 572 “no” cases and 578 “yes” cases, misclassifying 47 as false positives and 53 as false negatives. It has balanced performance but leans slightly towards more false negatives (53), which may reduce sensitivity for “yes” predictions.

Random Forest: Correctly predicts 562 “no” cases and 585 “yes” cases, with 40 false positives and 63 false negatives. This model has the fewest false positives, indicating it is more cautious about predicting “yes” but misclassifies slightly more “no” cases than C5.0

Neural Network: Correctly classifies 579 “no” cases and 578 “yes” cases, with 47 false positives and 46 false negatives. This model has the lowest total misclassifications (93), making it the most balanced and accurate based on its confusion matrix.

Thus, Neural Network and C5.0 have the two of the lowest misclassifications and thus are the most reliable.

CONCLUSION

Considering both the test accuracy and confusion matrix, the Neural Network remains the best choice. It has the highest test accuracy (0.9256), the fewest total misclassifications, and balanced precision and recall between “yes” and “no” classes. But considering the application of the forecasting model at hand, **the C5.0 model is the most suitable** because:

- **Comparable Performance:** Although slightly less accurate than Neural Networks (92% vs. 92.56% in test accuracy), C5.0 offers a good balance of performance and interpretability.

- **Transparent Decision-Making:** C5.0 uses a decision-tree structure that provides clear, human-readable rules, showing exactly how features like distance and prior work-from-home days lead to each decision.

- **Interpretability Advantage:** Unlike "black-box" models like Neural Networks, C5.0 allows stakeholders to understand the model’s decision-making logic, making it ideal for policy or compliance needs expanding its applications outside resource planning.

- **Feature Importance:** C5.0 ranks features by importance, helping teams see which attributes are most predictive for the work-from-home classification.

- **Practical Adaptability:** C5.0’s rule-based decisions can easily be translated into actionable insights for resource planning, which is often difficult with more complex models.

This makes C5.0 an excellent choice when transparency and actionable insights are priorities for the forecasting model.

THANK YOU!