# Guide d'implémentation du modèle Datagotchi de prédiction de vote

## 1. Introduction et fonctionnement général

Ce modèle permet de prédire les intentions de vote (BQ, CPC, GPC, LPC, NDP) à partir des caractéristiques socio-démographiques et des préférences de style de vie des utilisateurs.

### Principe général:

1. L'utilisateur répond à des questions (région, âge, préférences, etc.)
2. Chaque réponse est convertie en variables selon le mapping
3. Chaque variable est multipliée par son coefficient correspondant
4. Les valeurs sont additionnées pour calculer le "score" de chaque parti
5. Les scores sont convertis en probabilités via la fonction softmax
6. Une correction idéologique est appliquée pour améliorer la précision

### Formule de base:

```
score_parti = intercept_parti + (coef_var1_parti * var1) + (coef_var2_parti * var2) + ...
probabilité_parti = exp(score_parti) / somme(exp(score_tous_partis))
```

## 2. Comment traiter les variables

### Traitement des variables:

1. **Variables binaires (0/1)**:
   - Exemple: gender_female = 1 si l'utilisateur a choisi "Femme", sinon 0
   - Coefficient appliqué directement: coef * valeur
2. **Variables catégorielles**:
   - Une seule modalité est active (1), les autres sont inactives (0)
   - Exemple pour "dwelling": si l'utilisateur choisit "townhouse", alors dwelling_townhouse=1 et toutes les autres dwelling_X=0
   - La modalité de référence n'a pas de coefficient (valeur 0)
   - Pour "rule_region", la référence est "prairies" (coefficient = 0)
3. **Variables numériques**:
   - Pour "age", utilisez directement la valeur (entre 0-99)

- Pour les variables à échelle (smoking, food, etc.), utilisez la valeur indiquée dans le mapping

4. **Variables spéciales**:
   - Variables "rule_X": ces variables sont déterminées par des règles spécifiques
   - Pour "rule_immigrant", c'est 1 si l'utilisateur est immigrant, 0 sinon

## 3. Variables et interactions du modèle

Le modèle inclut de nombreuses variables et interactions entre ces variables. Ces interactions ont été sélectionnées systématiquement pour optimiser la performance prédictive du modèle.

### Formule complète du modèle:

```
dv_voteChoice ~ ses_region + ses_immigrant + lifestyle_typeTransport + lifestyle_consClothes
lifestyle_exercise + lifestyle_eatMeatFreq + lifestyle_favAlcool + lifestyle_consCoffee +
ses_language + lifestyle_smokeFreq + ses_age + ses_dwelling_cat + ses_ethnicityWhite +
ses_sexOrientationHetero + ses_genderFemale + lifestyle_clothingStyleGroups +
lifestyle_goHuntingFreq_numeric + lifestyle_goFishingFreq_bin + lifestyle_goMuseumsFreq_bin +
lifestyle_volunteeringFreq + lifestyle_motorizedActFreq_bin + lifestyle_hasTattoos +
ses_educ + ses_income3Cat + lifestyle_ownPet_bin +
lifestyle_consCoffee:ses_income3Cat +
lifestyle_consCoffee:lifestyle_goMuseumsFreq_bin +
lifestyle_favAlcool:lifestyle_goMuseumsFreq_bin +
ses_sexOrientationHetero:lifestyle_ownPet_bin +
lifestyle_goHuntingFreq_numeric:lifestyle_hasTattoos +
ses_dwelling_cat:lifestyle_hasTattoos +
ses_ethnicityWhite:lifestyle_goFishingFreq_bin +
ses_ethnicityWhite:ses_educ +
ses_sexOrientationHetero:ses_educ +
lifestyle_volunteeringFreq:lifestyle_motorizedActFreq_bin
```

### Les interactions sélectionnées (classées par ordre d'importance):

| Interaction | Score AIC |
|---|---|
| lifestyle_consCoffee:ses_income3Cat | 10937.35 |
| lifestyle_consCoffee:lifestyle_goMuseumsFreq_bin | 10952.93 |
| lifestyle_favAlcool:lifestyle_goMuseumsFreq_bin | 10956.35 |
| ses_sexOrientationHetero:lifestyle_ownPet_bin | 10971.62 |

| Interaction | Score AIC |
|---|---|
| lifestyle_goHuntingFreq_numeric:lifestyle_hasTattoos | 10978.35 |
| ses_dwelling_cat:lifestyle_hasTattoos | 10987.37 |
| ses_ethnicityWhite:lifestyle_goFishingFreq_bin | 10989.34 |
| ses_ethnicityWhite:ses_educ | 10989.94 |
| ses_sexOrientationHetero:ses_educ | 10990.07 |
| lifestyle_volunteeringFreq:lifestyle_motorizedActFreq_bin | 10995.01 |

Note: Le score AIC (Akaike Information Criterion) plus bas indique une meilleure performance prédictive.

**Explication des interactions:**

Les interactions (indiquées par le symbole `:`) représentent l'effet combiné de deux variables, qui peut être différent de la simple somme de leurs effets individuels. Par exemple:

1. **lifestyle_consCoffee:ses_income3Cat** (interaction la plus importante)
    - L'effet de la consommation de café sur l'intention de vote varie selon la catégorie de revenu
    - Une préférence pour le café de spécialité peut avoir un impact différent chez les personnes à revenu élevé vs. faible
2. **lifestyle_consCoffee:lifestyle_goMuseumsFreq_bin**
    - L'effet combiné de la consommation de café et de la fréquentation des musées
    - Capture potentiellement un certain profil culturel/intellectuel
3. **lifestyle_favAlcool:lifestyle_goMuseumsFreq_bin**
    - L'effet de la préférence pour certains types d'alcool varie selon la fréquentation des musées

**Traitement des interactions dans l'implémentation:**

1. **Calcul des termes d'interaction**:
    - Pour une interaction entre variables binaires: multipliez simplement les valeurs (0/1 $\times$ 0/1)
    - Pour une interaction avec une variable numérique: multipliez la valeur numérique par la valeur binaire ou catégorielle
    - Pour une interaction avec une variable catégorielle: créez des interactions distinctes pour chaque modalité

2. **Exemple Python pour le calcul des interactions**:

```python
def calculer_interactions(variables, coefficients, scores):
    # Définir toutes les interactions du modèle
    interactions = [
        "lifestyle_consCoffee:ses_income3Cat_high",
        "lifestyle_consCoffee:ses_income3Cat_medium",
        "lifestyle_consCoffee:lifestyle_goMuseumsFreq_bin",
        "lifestyle_favAlcool:lifestyle_goMuseumsFreq_bin",
        "ses_sexOrientationHetero:lifestyle_ownPet_bin",
        "lifestyle_goHuntingFreq_numeric:lifestyle_hasTattoos",
        "ses_dwelling_cat_apartment:lifestyle_hasTattoos",
        "ses_dwelling_cat_house:lifestyle_hasTattoos",
        "ses_dwelling_cat_townhouse:lifestyle_hasTattoos",
        "ses_ethnicityWhite:lifestyle_goFishingFreq_bin",
        "ses_ethnicityWhite:ses_educ_bachelor",
        "ses_ethnicityWhite:ses_educ_college",
        "ses_ethnicityWhite:ses_educ_graduate",
        "ses_ethnicityWhite:ses_educ_highschool",
        "ses_sexOrientationHetero:ses_educ_bachelor",
        "ses_sexOrientationHetero:ses_educ_college",
        "ses_sexOrientationHetero:ses_educ_graduate",
        "ses_sexOrientationHetero:ses_educ_highschool",
        "lifestyle_volunteeringFreq:lifestyle_motorizedActFreq_bin"
    ]

    for interaction in interactions:
        # Séparer l'interaction en ses composantes
        components = interaction.split(':')
        var1 = components[0]
        var2 = components[1]

        # Vérifier si les deux variables sont présentes
        if var1 in variables and var2 in variables:
            # Calculer la valeur de l'interaction
            interaction_value = variables[var1] * variables[var2]

            # Appliquer le coefficient pour chaque parti
            for parti in scores:
                if interaction in coefficients and parti in coefficients[interaction]:
                    scores[parti] += coefficients[interaction][parti] * interaction_value
```

## 4. Calcul des scores et probabilités

### Étapes de calcul en Python:

1. **Initialiser les scores**:

```
scores = {
    'bq': intercept_bq,
    'cpc': intercept_cpc,
    'gpc': intercept_gpc,
    'lpc': intercept_lpc,
    'ndp': intercept_ndp
}
```

2. **Appliquer les coefficients**: Pour chaque réponse de l'utilisateur:
   - Identifier la variable correspondante selon le mapping
   - Pour chaque parti, ajouter (coefficient * valeur) au score du parti

   Exemple Python:

```
def appliquer_coefficient(parti, variable, valeur, scores):
    if variable in coefficients and parti in coefficients[variable]:
        scores[parti] += coefficients[variable][parti] * valeur
```

3. **Convertir en probabilités (fonction softmax)**:

```
def softmax(scores):
    exp_scores = {parti: math.exp(score) for parti, score in scores.items()}
    total = sum(exp_scores.values())
    return {parti: exp_score/total for parti, exp_score in exp_scores.items()}
```

## 5. Correction idéologique

### Méthode de correction idéologique:

1. **Matrice de proximité**:
   - Chaque parti a un degré de "proximité" avec les autres partis
   - La diagonale = 1.0 (un parti est 100% proche de lui-même)
   - Exemple: proximité(BQ, NDP) = 0.5, proximité(CPC, GPC) = 0.3
2. **Poids idéologiques par parti**:

- Chaque parti a un poids spécifique qui détermine l'importance de la correction
- Optimisés pour maximiser la précision du modèle (0.5 pour tous les partis d'après les tests)

3. **Formule de correction**:

```
proba_corrigée(parti_A) = poids_parti_A * proba_originale(parti_A) +
                          (1-poids_parti_A) * somme(proba_originale(parti_B) * proximité(parti
```

Où la somme est calculée pour tous les partis B

4. **Implémentation en Python**:

```python
def appliquer_correction_ideologique(probas_originales, matrice_proximite, poids_partis):
    probas_corrigees = {}
    for parti_A in probas_originales:
        composante_originale = probas_originales[parti_A] * poids_partis[parti_A]
        composante_ideologique = 0
        for parti_B in probas_originales:
            composante_ideologique += probas_originales[parti_B] * matrice_proximite[parti_A]
        composante_ideologique *= (1 - poids_partis[parti_A])
        probas_corrigees[parti_A] = composante_originale + composante_ideologique

    # Renormaliser pour que la somme = 1
    total = sum(probas_corrigees.values())
    return {parti: proba/total for parti, proba in probas_corrigees.items()}
```

## 6. Matrice de proximité idéologique

Matrice de proximité entre partis (format CSV):

| parti | bq | cpc | gpc | lpc | ndp |
|---|---|---|---|---|---|
| bq | 1.0 | 0.2 | 0.3 | 0.4 | 0.5 |
| cpc | 0.2 | 1.0 | 0.3 | 0.5 | 0.3 |
| gpc | 0.3 | 0.3 | 1.0 | 0.6 | 0.7 |
| lpc | 0.4 | 0.5 | 0.6 | 1.0 | 0.7 |
| ndp | 0.5 | 0.3 | 0.7 | 0.7 | 1.0 |

## 7. Poids idéologiques par parti

Poids par parti (optimisés selon les tests):

| parti | poids_original |
|---|---|
| bq | 0.5 |
| cpc | 0.5 |
| gpc | 0.5 |
| lpc | 0.5 |
| ndp | 0.5 |

## 8. Exemple concret de calcul

### Exemple de calcul complet:

Supposons un utilisateur avec ces caractéristiques: - Femme (gender_female = 1) - Âge 35 ans (age = 35) - Région: Québec (rule_region_quebec = 1, autres regions = 0) - Transport: Voiture (transport_car = 1) - Éducation: Baccalauréat (education_bachelor = 1)

1. **Calcul des scores bruts**:

   ```
   score_bq = intercept_bq + (0.1216234291 * 1) + (0.01234633102 * 35) + (-1.0382126 * 1) +
   ... (faire de même pour chaque parti)
   ```

2. **Conversion en probabilités (softmax)**:

   ```
   probabilités = softmax(scores)
   ```

   Exemple: probabilité_bq = 0.35, probabilité_cpc = 0.15, probabilité_gpc = 0.10, probabilité_lpc = 0.25, probabilité_ndp = 0.15

3. **Application de la correction idéologique**:

   ```
   probabilités_corrigées = appliquer_correction_ideologique(probabilités, matrice_proximit
   ```

   Exemple de calcul pour le BQ:

   ```
   proba_corrigée_bq = 0.5 * 0.35 + 0.5 * (0.35*1.0 + 0.15*0.2 + 0.10*0.3 + 0.25*0.4 + 0.15
                     = 0.175 + 0.5 * (0.35 + 0.03 + 0.03 + 0.10 + 0.075)
                     = 0.175 + 0.5 * 0.585
                     = 0.175 + 0.2925
                     = 0.4675
   ```

   Puis renormaliser toutes les probabilités pour que leur somme = 1

## 9. Performances du modèle

Le modèle a été entraîné et évalué avec les résultats suivants:

- Accuracy du modèle sans correction idéologique: **49.52%**
- Accuracy du modèle avec correction idéologique: **51.11%**
- Amélioration: **1.59%**

### Performances par parti:

| Parti | Sans correction | Avec correction | Amélioration |
|---|---|---|---|
| BQ | 85.2% | 82.7% | -2.5% |
| CPC | 45.1% | 42.7% | -2.4% |
| GPC | 26.3% | 15.8% | -10.5% |
| LPC | 23.9% | 38.9% | +15.0% |
| NDP | 48.6% | 42.9% | -5.7% |

La correction idéologique améliore significativement les prédictions pour le LPC, au détriment léger des autres partis.

## 10. Conseils supplémentaires pour l'implémentation

1. Incluez l'onglet avec les intercepts pour chaque parti (à extraire du modèle R)
2. Incluez l'onglet avec les coefficients complets pour chaque variable et chaque parti
3. Ajoutez un onglet avec des exemples de requêtes d'API ou de code Python que les développeurs pourront utiliser comme point de départ
4. Préparez un petit ensemble de données de test avec des entrées et des sorties attendues pour qu'ils puissent valider leur implémentation